MUSIC LIBRARY MANAGEMENT SYSTEM

Introduction to database project

****

**AMERICAN INTERNATIONAL UNIVERSITY BANGLADESH**

**(AIUB)**

**FACULTY OF SCIENCE & TECHNOLOGY**

**DEPARTMENT OF CSE**

**Project report on**

**Music Library management system**

**Supervised By**

**Supta Richard Philip**

**Submitted By**

|  |  |  |
| --- | --- | --- |
| NAME | ID | CONTRIBUTION |
| Saif Ahmed Ayon | 22-47572-2 | Intro+Case study |
| Sourav Saha | 22-47911-2 | ER diagram+Normalization |
| Md.Abdullah Al Noman | 22-47844-2 | Finalization+Table description |
| Ujjoyeni Dey | 22-49001-3 | Table data insertion+Query+conclusion |

Contents

[1. Introduction 2](#_Toc141598762)

[2. Case Study 3](#_Toc141598763)

[3. ER Diagram 4](#_Toc141598764)

[4. Normalization .5](#_Toc141598765)-7

5. Finalization…………………………………………………………………………………8-9

6. Table Description……………………………………………………………………..….10-16

[7. Table Data Insertion 17-23](#_Toc141598768)

[8. Query 24-31](#_Toc141598769)

[9. Conclusion](#_Toc141598770) 32

# Introduction

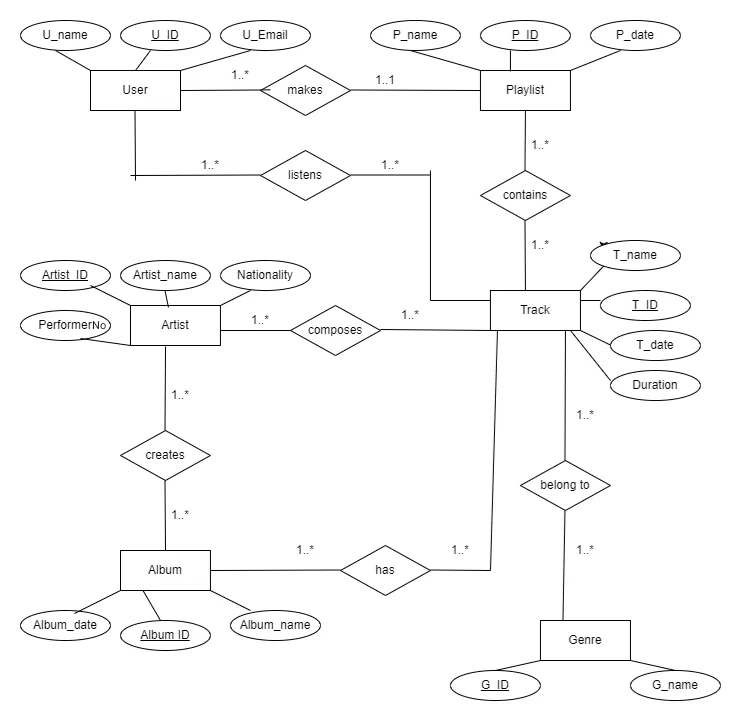
A music library management system in a database is a software application that helps music library management and organize various aspects of their operations. The music database management project is to categorize and catalog every single piece of music for music info and reviews, review albums, listing record collections. The idea of music database arose out of common interest of project member in music. The concept seemed different and very interesting right in the first go.

# Case Study

# MUSIC LIBRARY MANAGEMENT SYSTEM CASE STUDY

In a music library management system, a user can listen to tracks/songs. A user can listen to many tracks and one track can also be listened to by many users. The system stores the  
user id (primary key), username and user email. A track is identified by the track id, track name, track creation date, and duration. Users can make playlists. One user can make many playlists, but one playlist can only be made by exactly one user. The playlist contains tracks. One playlist can contain many tracks and one track can be in many playlists. The system stores the playlist id, playlist name, and playlist created date. Artist  
composes track. A track may be composed by many artists and one artist can compose many tracks. The system stores the artist id, artist name, performer and nationality, of the artists who  
created at least one track. An artist creates an Album. An album may be created by many artists and one artist can create many albums. Albums are identified by album id, album name, and album date created. One artist has to create at least one track or an album. Album has tracks. One album may contain many tracks and one track can only belong to one album. Tracks belong to genres. One track can belong to multiple genres and for one genre  
there can be multiple tracks. Genre is specified by genre id, and genre name

# ER Diagram



# Normalization

UNF (makes): U\_ID, U\_Name, U\_Email, P\_ID, P\_Name, P\_Date

1NF: **U-ID**, U\_Name, U\_Email,

**P-ID**, P\_Name, P\_Date

2NF: 1) **U-ID**, U\_Name, U\_Email, P\_ID (FK)

2) **P-ID**, P\_Name, P\_Date

3NF: Same as 2NF

UNF (contains): P\_ID, P\_Name, P\_Date, T\_Date, T\_Name, T\_ID, Duration

1NF: **P-ID**, P\_Name, P\_Date

**T-ID**, T\_Name, T\_Date, Duration

2NF: 1) **P-ID**, P\_Name, P\_Date

2) **T-ID**, T\_Name, T\_Date, Duration

3) P\_ID (PK), T\_ID (FK) / P\_ID (FK), T\_ID (PK)

3NF: Same as 2NF

UNF (listens): U\_ID, U\_Name, U\_Email, T\_Date, T\_Name, T\_ID, Duration

1NF: **U-ID**, U\_Name, U\_Email

**T-ID**, T\_Date, T\_Name, Duration

2NF: 1) **U-ID**, U\_Name, U\_Email

2) **T-ID**, T\_Date, T\_Name, Duration

3) U\_ID (PK), T\_ID (FK) / U\_ID (FK), T\_ID (PK)

3NF: Same as 2NF

UNF (belongs): T\_Date, T\_Name, T\_ID, Duration, G\_ID, G\_Name

1NF: **T-ID**, T\_Date, T\_Name, Duration, **G-ID**, G\_Name

2NF: 1) **T-ID**, T\_Date, T\_Name, Duration

2) **G-ID**, G\_Name

3) T\_ID (PK), G\_ID (FK) / T\_ID (FK), G\_ID (PK)

3NF: Same as 2NF

UNF (composes): Artist\_ID, Artist\_Name, Nationality, PerformerNo, T\_Date, T\_ID, T\_Name, Duration

1NF: **Artist-ID**, Artist\_Name, Nationality , PerformerNo

**T-ID**, T\_Date, T\_Name, Duration

2NF: 1) **Artist-ID**, Artist\_Name, Nationality, PerformerNo

2) **T-ID** ,T\_Date, T\_Name, Duration

3) Artist\_ID (PK), T\_ID (FK) / Artist\_ID (FK), T\_ID (PK)

3NF: Same as 2NF

UNF (creates): Artist\_ID, Artist\_Name, Nationality, PerformerNo, Album\_ID, Album\_Name, Album\_date

1NF: **Artist-ID**, Artist\_Name, Nationality, PerformerNo

**Album-ID**, Album\_Name, Album\_date

2NF: 1) **Artist-ID**, Artist\_Name, Nationality, PerformerNo

2) **Album-ID**, Album\_Name, Album\_date

3) Artist\_ID (PK), Album\_ID (FK) / Artist\_ID(FK), Album\_ID (PK)

3NF: Same as 2NF

UNF (has): Album\_ID, Album\_Name, Album\_Date, T\_Date, T\_Name, T\_ID, Duration

1NF: **Album-ID**, Album\_Name, Album\_Date

**T-ID**, T\_Name, T\_Date, Duration

2NF: 1) **Album-ID**, Album\_Name, Album\_Date, T\_ID (FK)

2) **T-ID**, T\_Name, T\_Date, Duration

3NF: Same as 2NF

# Finalization

1) U-ID, U\_Name, U\_Email, P\_ID (FK)

2) P-ID, P\_Name, P\_Date

~~3) P-ID, P\_Name, P\_Date~~

4) T-ID, T\_Name, T\_Date, Duration

5) P\_ID (PK), T\_ID (FK)

6) U-ID, U\_Name, U\_Email

~~7) T-ID, T\_Date, T\_Name, Duration~~

8) U\_ID (PK), T\_ID (FK)

~~9) T-ID, T\_Date, T\_Name, Duration~~

10) G-ID, G\_Name

11) T\_ID (PK), G\_ID (FK)

12) Artist-ID, Artist\_Name, Nationality, PerformerNo

~~13) T-ID ,T\_Date, T\_Name, Duration~~

14) Artist\_ID (PK), T\_ID (FK)

~~15) Artist-ID, Artist\_Name, Nationality,~~ ~~PerformerNo~~

16) Album-ID, Album\_Name, Album\_date

17) Artist\_ID (PK), Album\_ID (FK)

18) Album-ID, Album\_Name, Album\_Date, T\_ID (FK)

~~19) T-ID, T\_Name, T\_Date, Duration~~

## **Final Tables**

1)**U-ID**, U\_Name, U\_Email

2) **P-ID**, P\_Name, P\_Date

3) **U-ID**, U\_Name, U\_Email, P\_ID (FK)

4) **T-ID**, T\_Name, T\_Date, Duration

5 **Artist-ID**, Artist\_Name, Nationality

6)**Album-ID**, Album\_Name, Album\_Date

7) **Album-ID**, Album\_Name, Album\_Date, T\_ID (FK)

8) **G-ID**, G\_Name

9) P\_ID (PK), T\_ID (FK)

10) U\_ID (PK), T\_ID (FK)

11) T\_ID (PK), G\_ID (FK)

12) Artist\_ID (PK), T\_ID (FK)

13) Artist\_ID (PK), Album\_ID (FK)

**Total tables: 13**

# Table Description

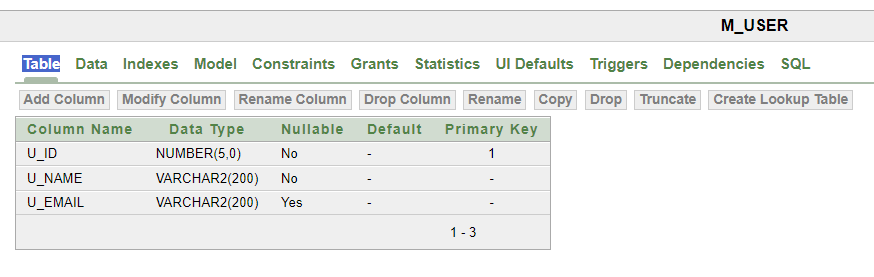
1) CREATE TABLE m\_user (

U\_ID number(5) primary key NOT NULL,

U\_Name varchar2(200) NOT NULL,

U\_Email varchar2(200)

);



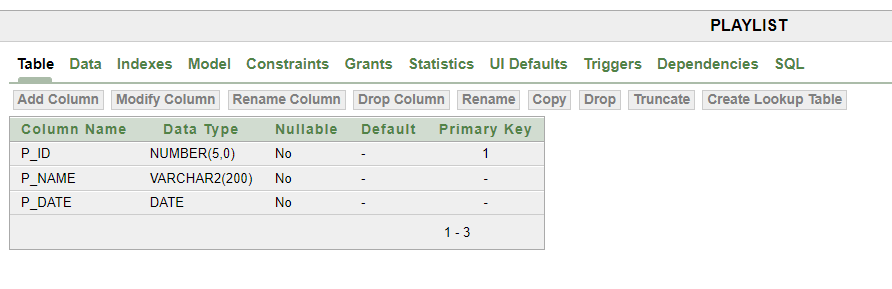
2) CREATE TABLE playlist (

P\_ID number (5) PRIMARY KEY NOT NULL,

P\_Name varchar2(200) NOT NULL,

P\_Date date NOT NULL

);



3) CREATE TABLE tracks (

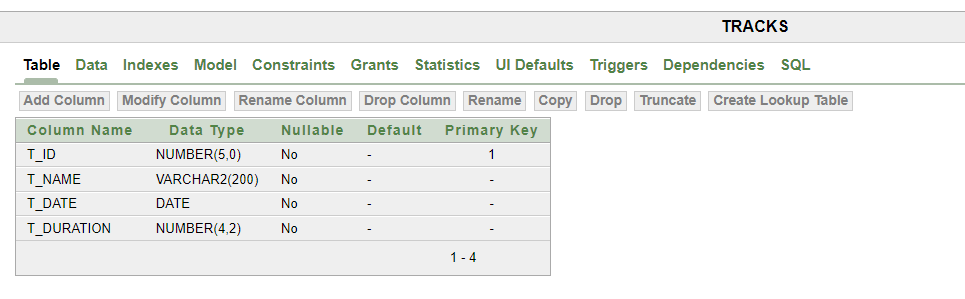
T\_ID number (5) PRIMARY KEY NOT NULL,

T\_Name varchar2(200) NOT NULL,

T\_Date date NOT NULL,

T\_Duration number(4,2) NOT NULL

);



4) CREATE TABLE artist (

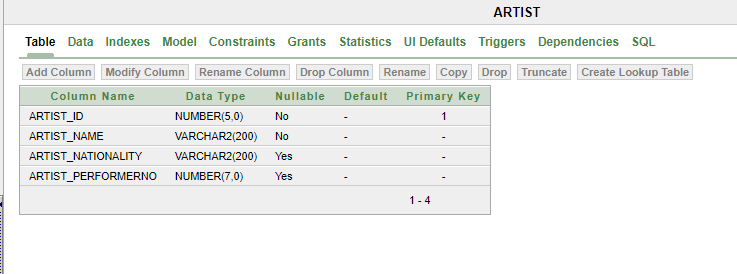
Artist\_ID number (5) PRIMARY KEY NOT NULL,

Artist\_Name varchar2(200) NOT NULL,

Artist\_Nationality varchar2(200),

Artist\_PerformerNo varchar2(200),

);



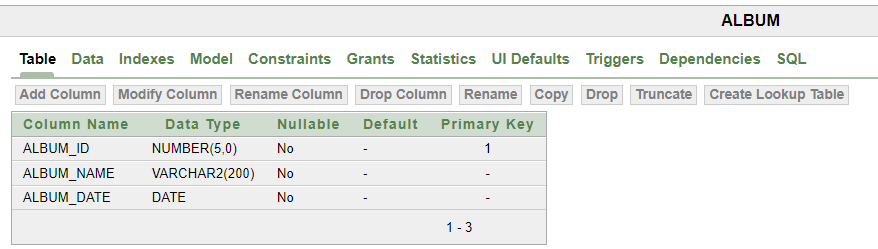
5) CREATE TABLE album (

Album\_ID number (5) PRIMARY KEY NOT NULL,

Album\_Name varchar2(200) NOT NULL,

Album\_Date date NOT NULL

);

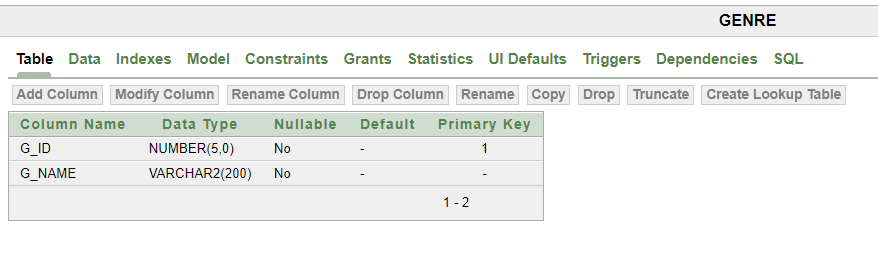


6) CREATE TABLE genre (

G\_ID number (5) PRIMARY KEY NOT NULL,

G\_Name varchar2(200) NOT NULL

);



7) CREATE TABLE makes (

U\_ID number(5) primary key NOT NULL,

U\_Name varchar2(200) NOT NULL,

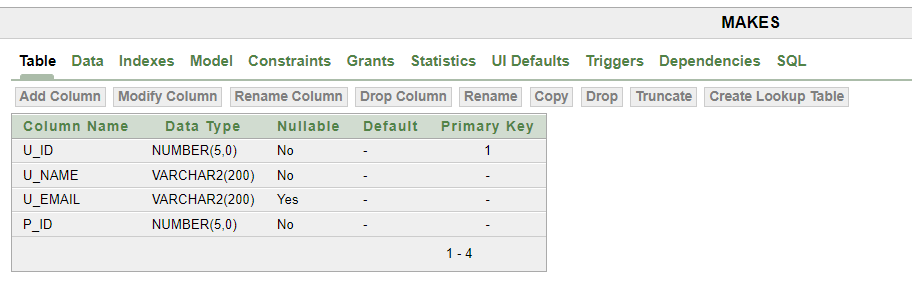
U\_Email varchar2(200),

P\_ID number(5) NOT NULL,

CONSTRAINT P\_ID FOREIGN KEY (P\_ID)

REFERENCES Playlist(P\_ID)

);



8) CREATE TABLE listens (

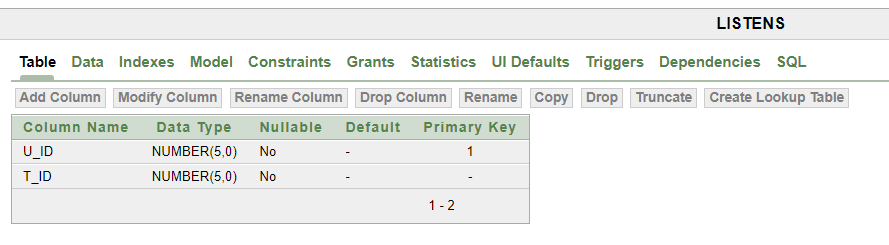
U\_ID number(5) PRIMARY KEY NOT NULL,

T\_ID number(5) NOT NULL,

CONSTRAINT tid4 FOREIGN KEY (T\_ID)

REFERENCES tracks(T\_ID)

);



9) CREATE TABLE contains (

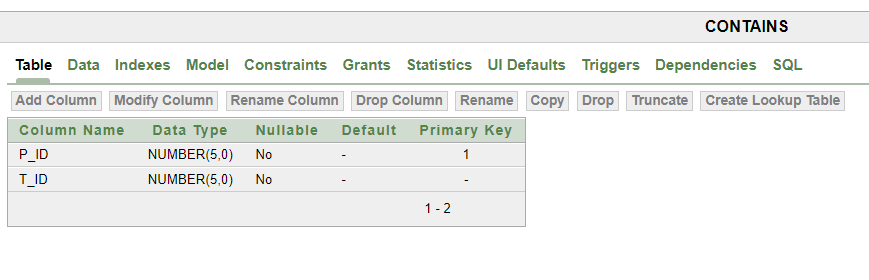
P\_ID number(5) PRIMARY KEY NOT NULL,

T\_ID number(5) NOT NULL,

CONSTRAINT tid FOREIGN KEY (T\_ID)

REFERENCES tracks(T\_ID)

);



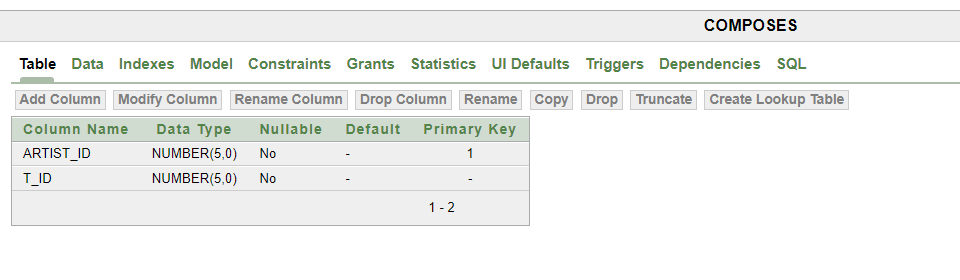
10) CREATE TABLE composes (

Artist\_ID number(5) PRIMARY KEY NOT NULL,

T\_ID number(5) NOT NULL,

CONSTRAINT tid3 FOREIGN KEY (T\_ID)

REFERENCES tracks(T\_ID)

); 

11) CREATE TABLE creates (

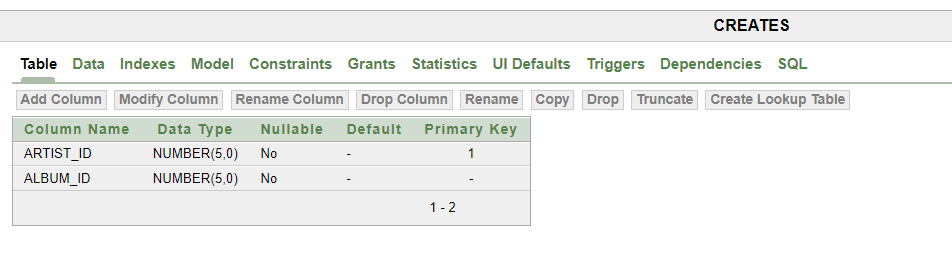
Artist\_ID number(5) PRIMARY KEY NOT NULL,

Album\_ID number(5) NOT NULL,

CONSTRAINT albumid FOREIGN KEY (Album\_ID)

REFERENCES album(Album\_ID)

);



12) CREATE TABLE has (

Album\_ID number (5) PRIMARY KEY NOT NULL,

Album\_Name varchar2(200) NOT NULL,

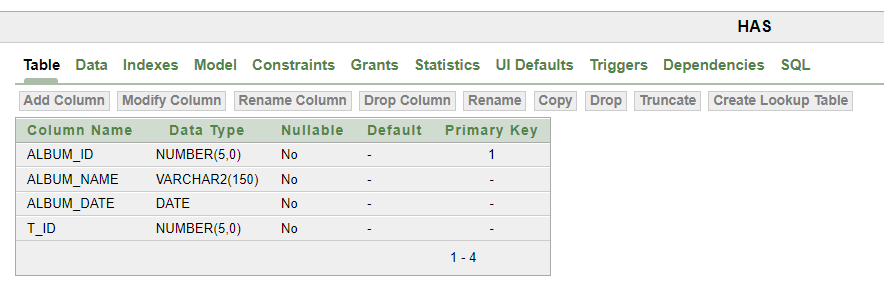
Album\_Date date NOT NULL,

T\_ID number (5) NOT NULL,

CONSTRAINT tid2 FOREIGN KEY (T\_ID)

REFERENCES tracks(T\_ID)

);



13) CREATE TABLE belongs (

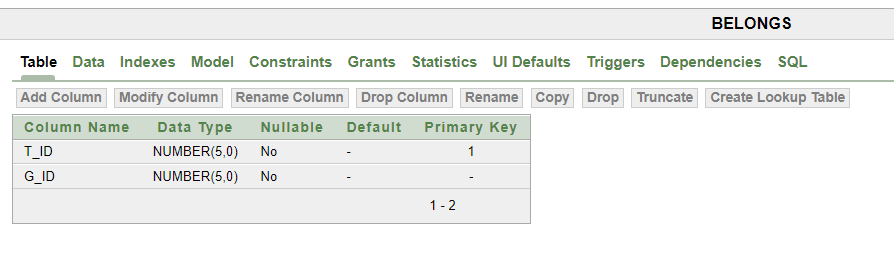
T\_ID number(5) PRIMARY KEY NOT NULL,

G\_ID number(5) NOT NULL,

CONSTRAINT gid FOREIGN KEY (G\_ID)

REFERENCES genre(G\_ID)

);



**7. Table Data Insertion**

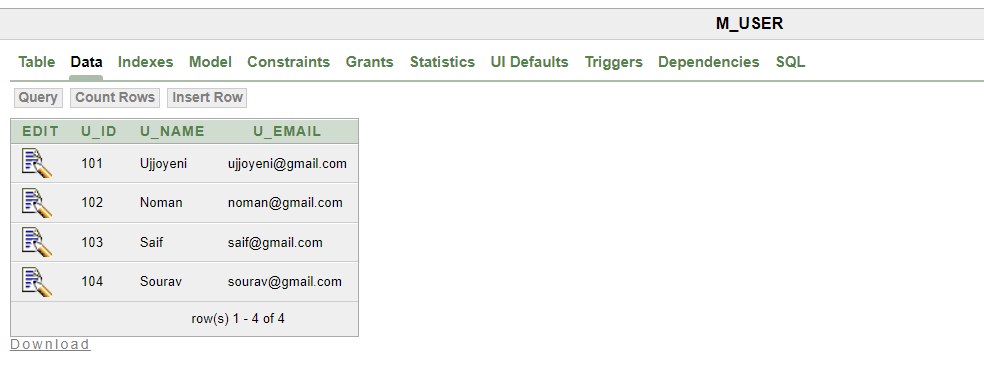
1) For m\_user table :

INSERT into m\_user values(101, 'Ujjoyeni' , 'ujjoyeni@gmail.com');

INSERT into m\_user values(102, 'Noman' , 'noman@gmail.com');

INSERT into m\_user values(103, 'Saif' , 'saif@gmail.com');

INSERT into m\_user values(104, 'Sourav' , 'sourav@gmail.com');



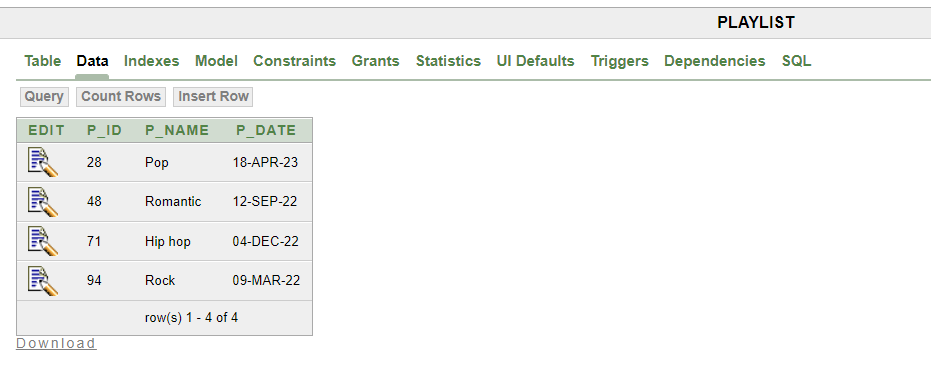
2) For playlist table :

INSERT into playlist values(28, 'Pop' , ‘18-Apr-23’);

INSERT into playlist values(48, 'Romantic' , ‘12-Sep-22’);

INSERT into playlist values(71, ' Hip hop' , ‘4-Dec-22’);

INSERT into playlist values(94, ' Rock' , ‘9-Mar-22’);



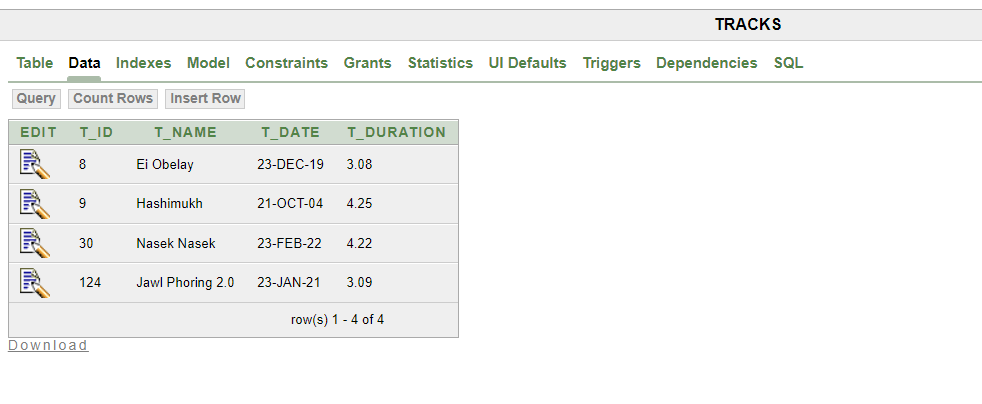
3) For tracks table :

INSERT into tracks values(8, 'Ei Obelay' , '23-Dec-19' , 3.08);

INSERT into tracks values(9, 'Hashimukh' , '21-Oct-04' , 4.25);

INSERT into tracks values(30, ' Nasek Nasek' , '23-Feb-22', 4.22);

INSERT into tracks values(124, 'Jawl Phoring 2.0' , ‘9-Mar-22’,3.08);



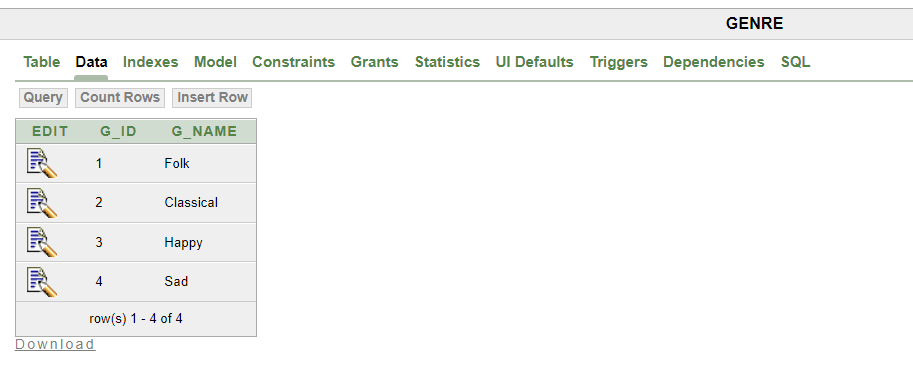
4) For genre table :

INSERT into genre values(1, 'Folk');

INSERT into genre values(2, 'Classical');

INSERT into genre values(3, 'Happy');

INSERT into genre values(4, 'Sad');



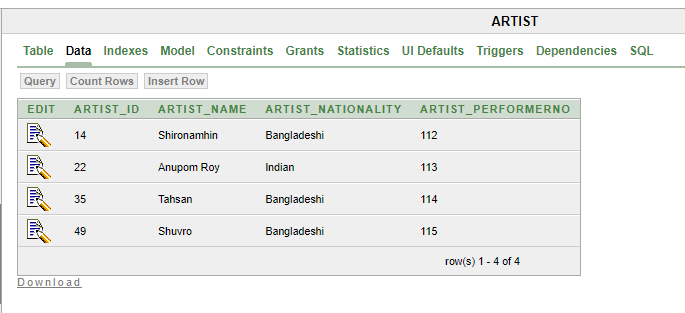
5) For artist table :

INSERT into artist values(14, 'Shironamhin', 'Bangladeshi',112);

INSERT into artist values(22, 'Anupom Roy', ‘Indian’113);

INSERT into artist values(35, 'Tahsan', ‘Bangladeshi’114);

INSERT into artist values(49, 'Shuvro', 'Bangladeshi'115);



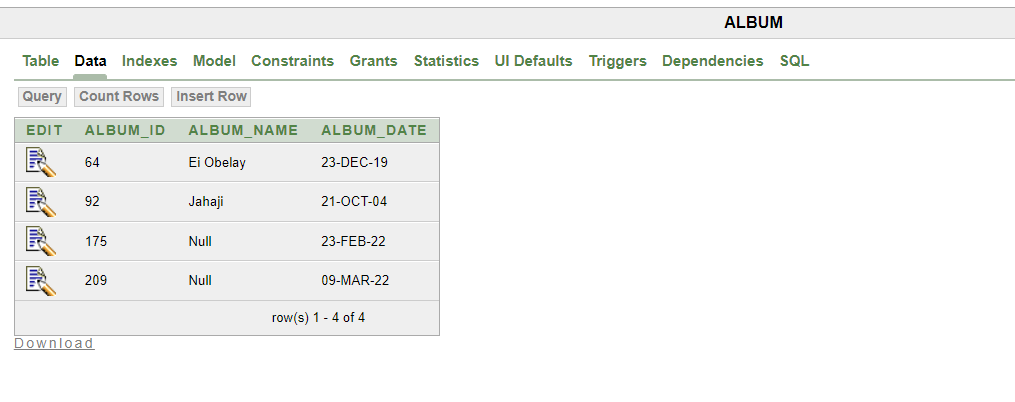
6) For album table :

INSERT into album values(64, 'Ei Obelay', '23-Dec-19');

INSERT into album values(92, 'Jahaji', '21-Oct-04');

INSERT into album values(175, 'Null', '23-Feb-22');

INSERT into album values(209, 'Null', '9-Mar-22');



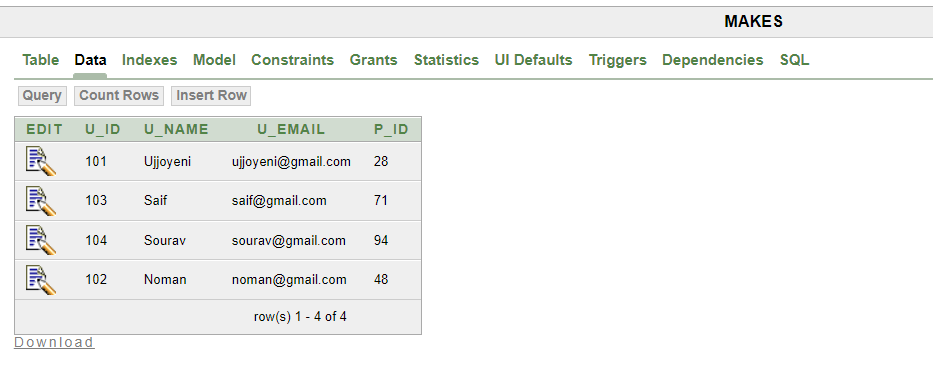
7) For makes table :

INSERT into makes values(101, 'Ujjoyeni', 'ujjoyeni@gmail.com', 28);

INSERT into makes values(102, 'Noman', 'noman@gmail.com',48);

INSERT into makes values(103, 'Saif', 'saif@gmail.com',71);

INSERT into makes values(104, 'Sourav', 'sourav@gmail.com',94);



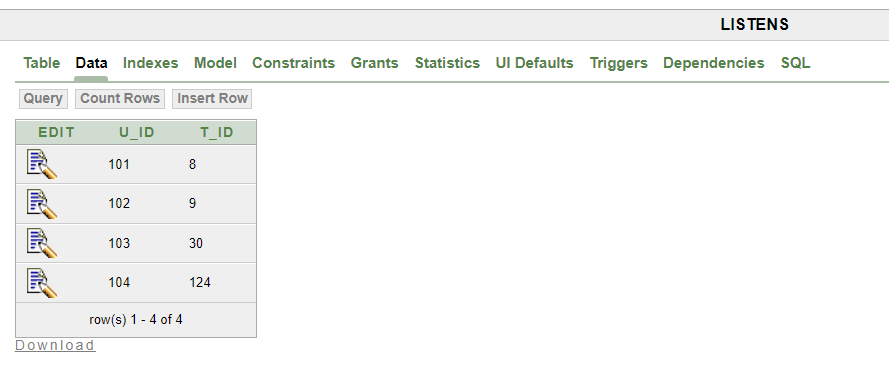
8) For listens table:

INSERT into listens values(101, 8);

INSERT into listens values(102, 9);

INSERT into listens values(103, 30);

INSERT into listens values(104, 124);



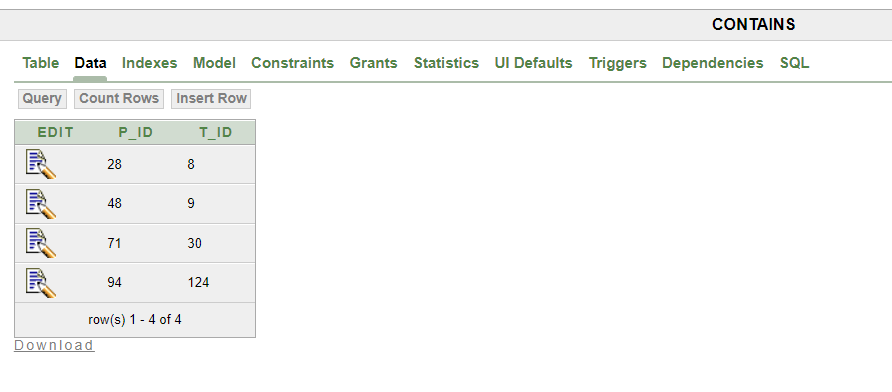
9) For contains table :

INSERT into contains values(28, 8);

INSERT into contains values(48, 9);

INSERT into contains values(71, 30);

INSERT into contains values(94, 124);



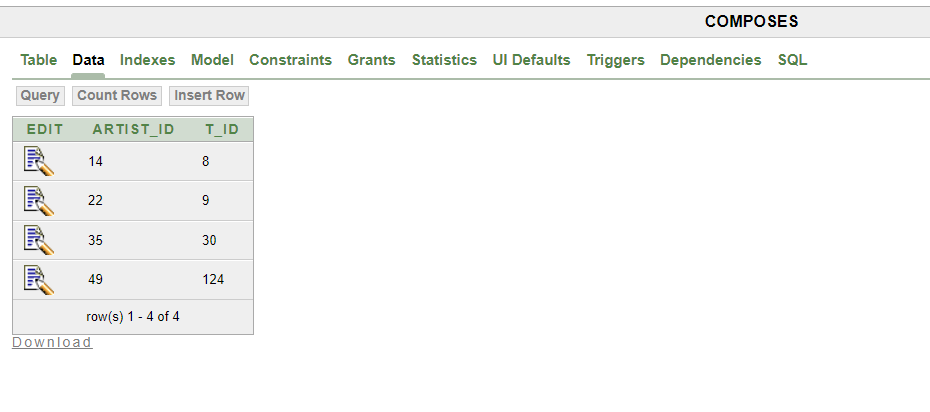
10) For composes table :

INSERT into composes values(14, 8);

INSERT into composes values(22, 9);

INSERT into composes values(35, 30);

INSERT into composes values(49, 124);



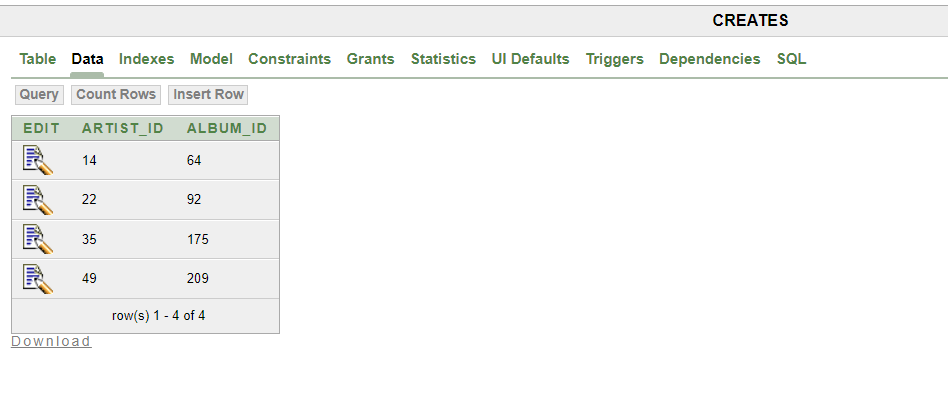
11) For creates table :

INSERT into creates values(14, 64);

INSERT into creates values(22, 92);

INSERT into creates values(35, 175);

INSERT into creates values(49, 209);



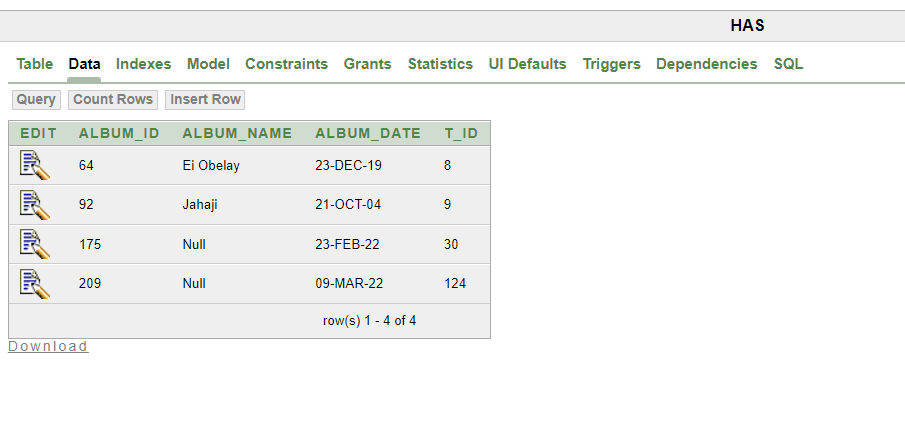
12) For has table :

INSERT into has values(64, 'Ei Obelay' , '23-Dec-19',8);

INSERT into has values(92, 'Jahaji', '21-Oct-04', 9);

INSERT into has values(175, 'Null', '23-Feb-22', 30);

INSERT into has values(209, 'Null', '9-Mar-22',124);



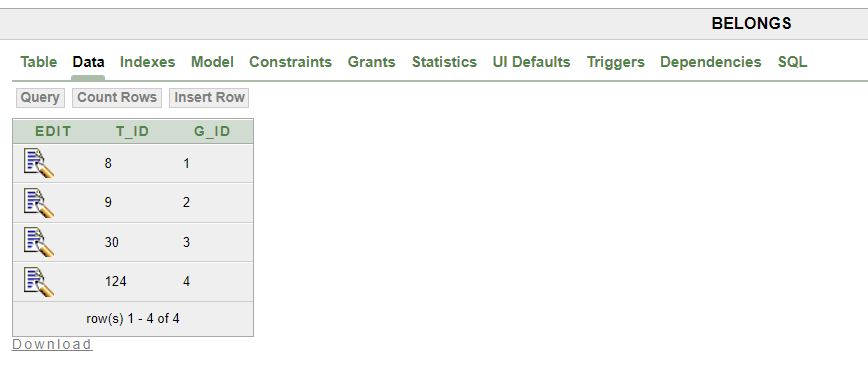
13) For belongs table :

INSERT into belongs values(8, 1);

INSERT into belongs values(9,2);

INSERT into belongs values(30, 3);

INSERT into belongs values(124, 4);



# 8. Query

***Single Row Subquery*** :

1. Show the track names with duration less than that of the track, 'Hashimukh' :

SELECT t\_name, t\_duration

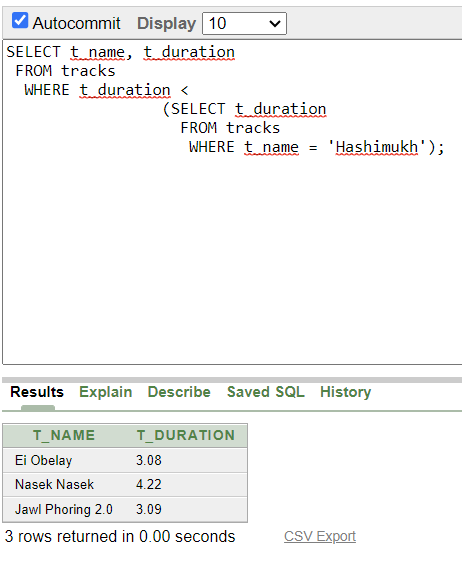
FROM tracks

WHERE t\_duration <

(SELECT t\_duration

FROM tracks

WHERE t\_name = 'Hashimukh');



2) Show the album names and creation date that was created before t\_id 30:

SELECT album\_name, album\_date

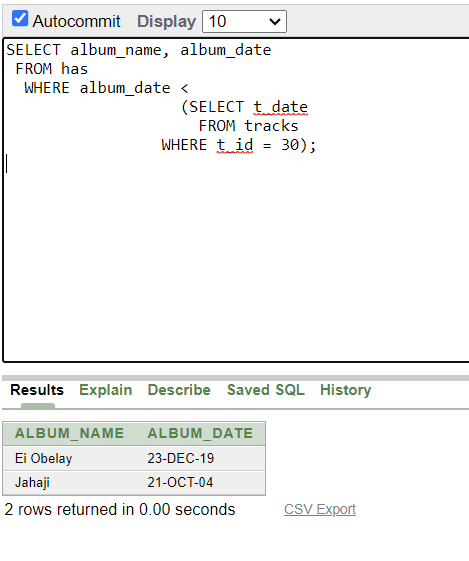
FROM has

WHERE album\_date <

(SELECT t\_date

FROM tracks

WHERE t\_id = 30);



***Multiple Row Subquery*** :

1. Show the artists who have tracks with the shortest duration:

SELECT artist\_name

FROM artist

WHERE artist\_id IN (

SELECT artist\_id

FROM tracks

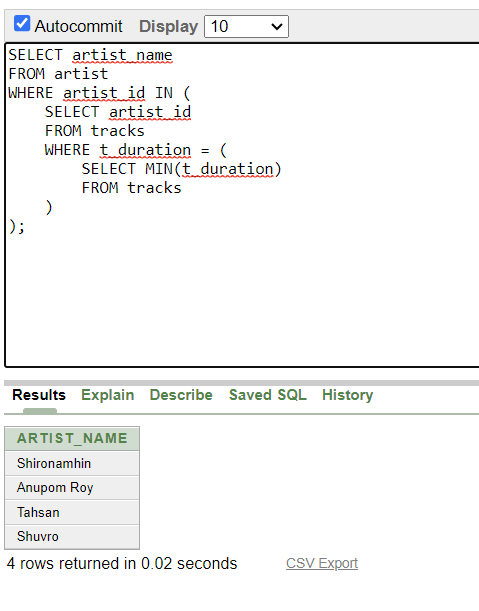
WHERE t\_duration = (

SELECT MIN(t\_duration)

FROM tracks

)

);



4) Show the user names who created a playlist after 9-Mar-22 :

SELECT u\_name

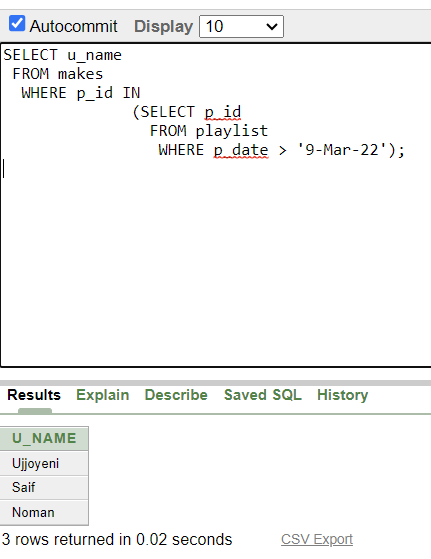
FROM makes

WHERE p\_id IN

(SELECT p\_id

FROM playlist

WHERE p\_date > '9-Mar-22');

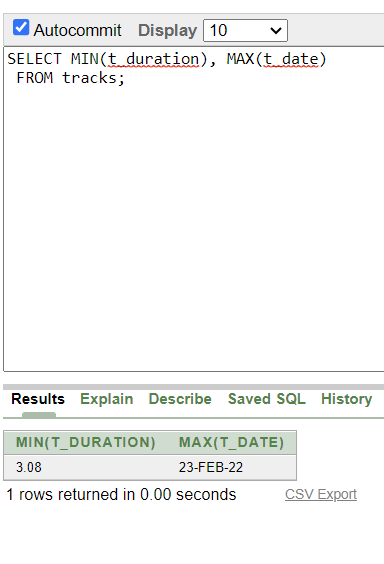


***Aggregate Function*** :

5) Show the minimum duration and the date of the last created track :

SELECT MIN(t\_duration), MAX(t\_date)

FROM tracks;



***Equijoin*** :

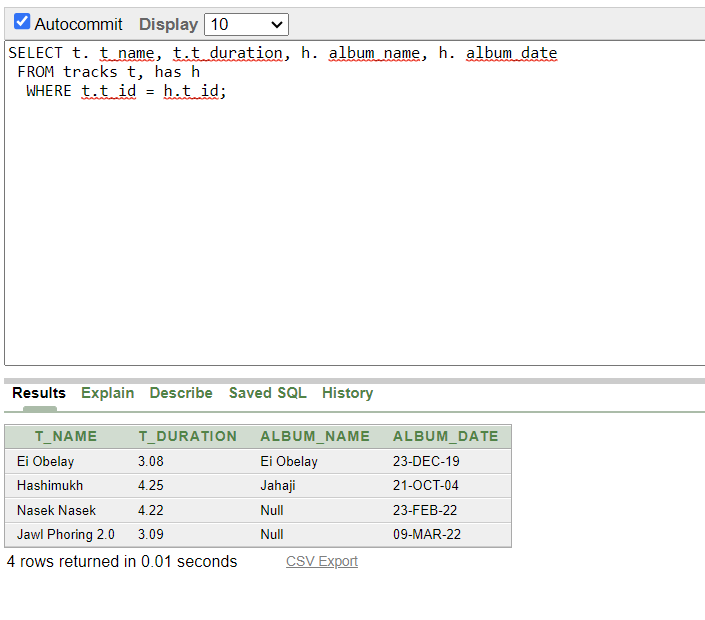
6) Show the track name, track duration, album name, album creation date

with proper joining condition :

SELECT t. t\_name, t.t\_duration, h. album\_name, h. album\_date

FROM tracks t, has h

WHERE t.t\_id = h.t\_id;



***Outer Join*** :

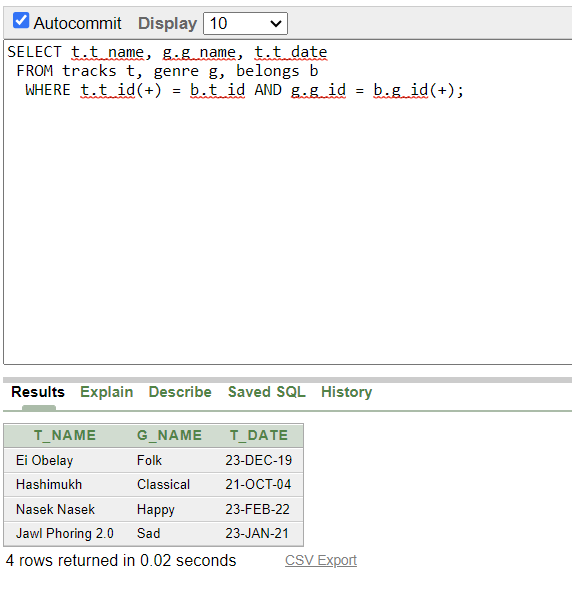
7) Show the track name, track creation date, genre name

with joining condition :

SELECT t.t\_name, g.g\_name, t.t\_date

FROM tracks t, genre g, belongs b

WHERE t.t\_id(+) = b.t\_id AND g.g\_id = b.g\_id(+);



# 9. Conclusion

In conclusion, a music library management system that utilizes a database using SQL can greatly improve the efficiency and effectiveness of managing a music library. The music Library Management System is much more user-friendly, faster in operation and easy to manage than the manual one. Through the use of it, the librarian can manage the whole data of the library in a single database in different tables with a much more security than the traditional way. Overall, implementing a music library management system using a database with SQL can provide numerous benefits and streamline operations for library industry.