**A Project Report on**

**Music Streaming Service**

**HERTZ**

DEVELOPED BY:

**IT034 Soumyadeep Ghosh**

**IT038 Niken Goswami**

**IT042 Dishank Inani**

**IT044 Naamsukh Jobanputra**

Guided By

Internal Guide:

Prof. Archana N. Vyas

Department of Information Technology

Faculty of Technology

DD University

****

**Department of Information Technology Faculty of Technology, Dharmsinh Desai University College Road, Nadiad-387001**

**October-2021**

**DHARMSINH DESAI UNIVERSITY**

**NADIAD-387001, GUJARAT**

****

**CERTIFICATE**

This is to certify that the project entitled “Music Streaming system” is

a bonafide report of the work carried out by

1. **Soumyadeep Ghosh** Student ID No: **19ITUOS094**

2) **Niken Goswami** Student ID No: **19ITUBS059**

3) **Naamsukh Jobanputra**  Student ID No: **19ITUOS118**

4) **Dishank Inani** Student ID No: **19ITUOS107**

of Department of Information Technology, semester V, under the guidance and

supervision for the subject Database Management System. They were involved in Project training during the academic year 2021-2022.

Prof. Archana N. Vyas

Project Guide, Department of Information Technology,

Faculty of Technology,

Dharmsinh Desai University, Nadiad

Date: 21/10/2021

Prof. Vipul Dabhi

Head, Department of Information Technology

**COMMENDATION**

We would like to express our heartfelt gratitude to everyone who contributed to the successful completion of our project "Hertz."

The success and ultimate conclusion of this project necessitated a great deal of advice and support from a large number of individuals, and we are incredibly fortunate to have received it all along with the project's completion.

We owe a debt of appreciation to Prof. Archana N. Vyas, our project guide, who took an interest in our project work and directed us through it till it was completed by giving all of the required assistance for creating a solid Database System.

We'd also want to express our gratitude to all of our speakers. Finally, we express our gratitude to all of our friends and colleagues.

**INDEX**

I.Certificate..............................................................................................................I

II. Commendation............................................................................................. II

1. SYSTEM OVERVIEW .........................................................................................5

1.1 Current system.....................................................................................................5

1.2 Objectives of the Proposed System ...................................................................6

1.3 Advantages of the Proposed system (over current) ............................................6

2. E-R DIAGRAM....................................................................................................7

3. SCHEMA DIAGRAM..........................................................................................8

4. DATA DICTIONARY..........................................................................................9

5. DATABASE IMPLEMENTATION...........................................................................15

5.1 Create Schema .....................................................................................................15

5.2 Insert Data values...........................................................................................20

5.3 Queries (Based on basic DBMS constructs) .................................................25

5.4 Queries (Based on Joins & Sub-Queries)………………………………………………..28

5.5 PL/SQL Blocks (Views) .................................................................................31

5.6 Functions & Triggers...............................................................................................32

5.7 Cursors……………………………………………………………………………………..35

6. FUTURE ENHANCEMENTS OF THE SYSTEM...........................................36

7. BIBLIOGRAPHY...............................................................................................37

**Source Code -** [**https://github.com/orgs/AOTitans/repositories**](https://github.com/orgs/AOTitans/repositories)

1. **SYSTEM OVERVIEW**

**1.1 CURRENT SYSTEM**

Our database will be designed like the Spotify streaming service, with artists, songs, categories, users, user playlists, and so on. Our major focus will be on structuring this data in such a way that playlists can be stored and a recommendation system for our users can be implemented later. It will also be done in an effective manner since we do not want our database to become redundant.

Within our platform, we want to encourage users to communicate with one another. The development of a 'collaboration' and 'following' connection inside our database addresses this aim.

**1.2 OBJECTIVES OF THE PROPOSED SYSTEM**

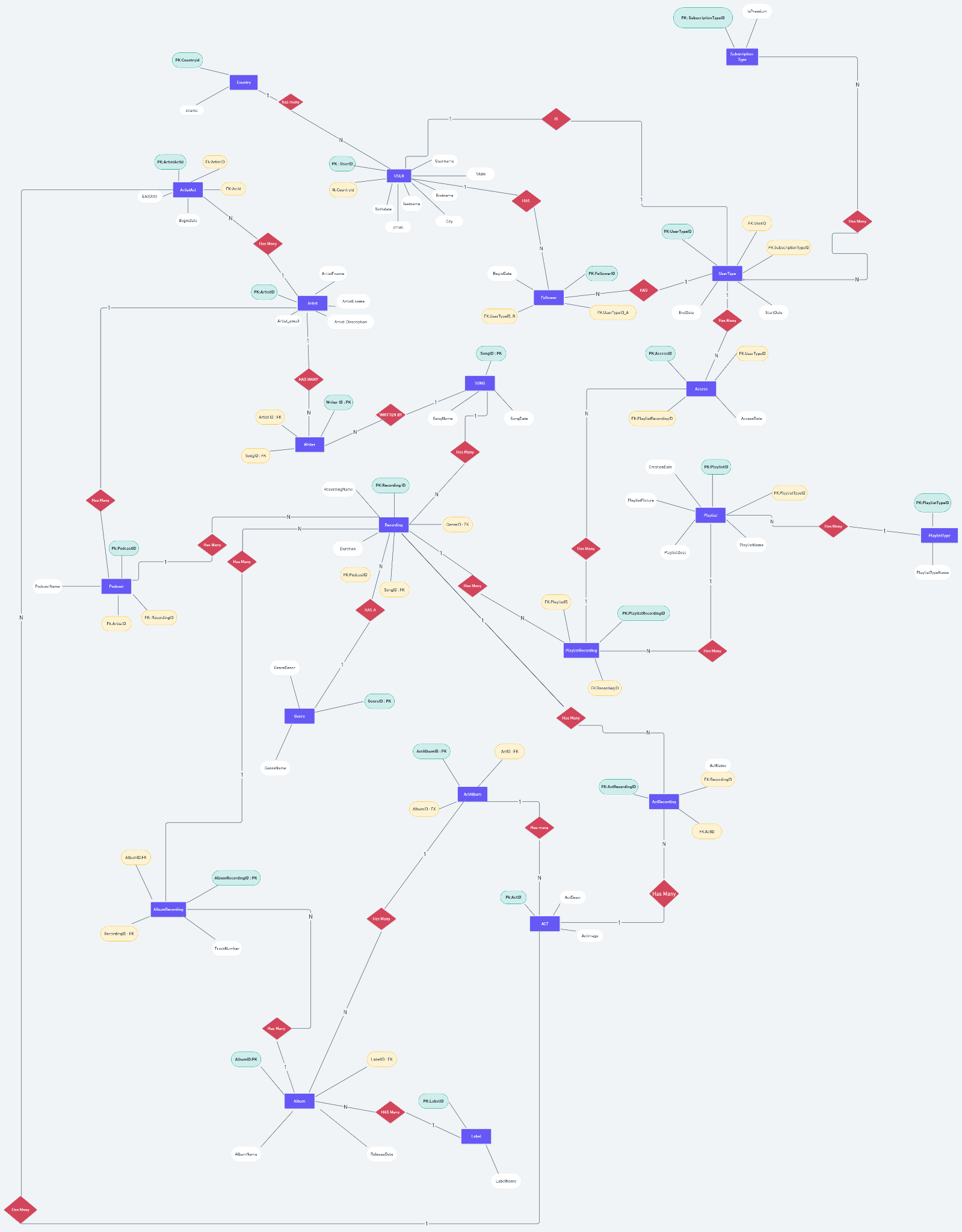
Music has progressed from being limited to tangible media to being widely available through digital methods during the last two decades. As a result, the method of music purchase has changed, moving from a pay-per-song approach to a more contemporary streaming strategy.

Our Project intends to address these two trends while also enhancing the user experience by offering tailored suggestions, a social network, and 24/7 access with the option to listen to music on the move.

**1.3 ADVANTAGES OF THE PROPOSED SYSTEM**

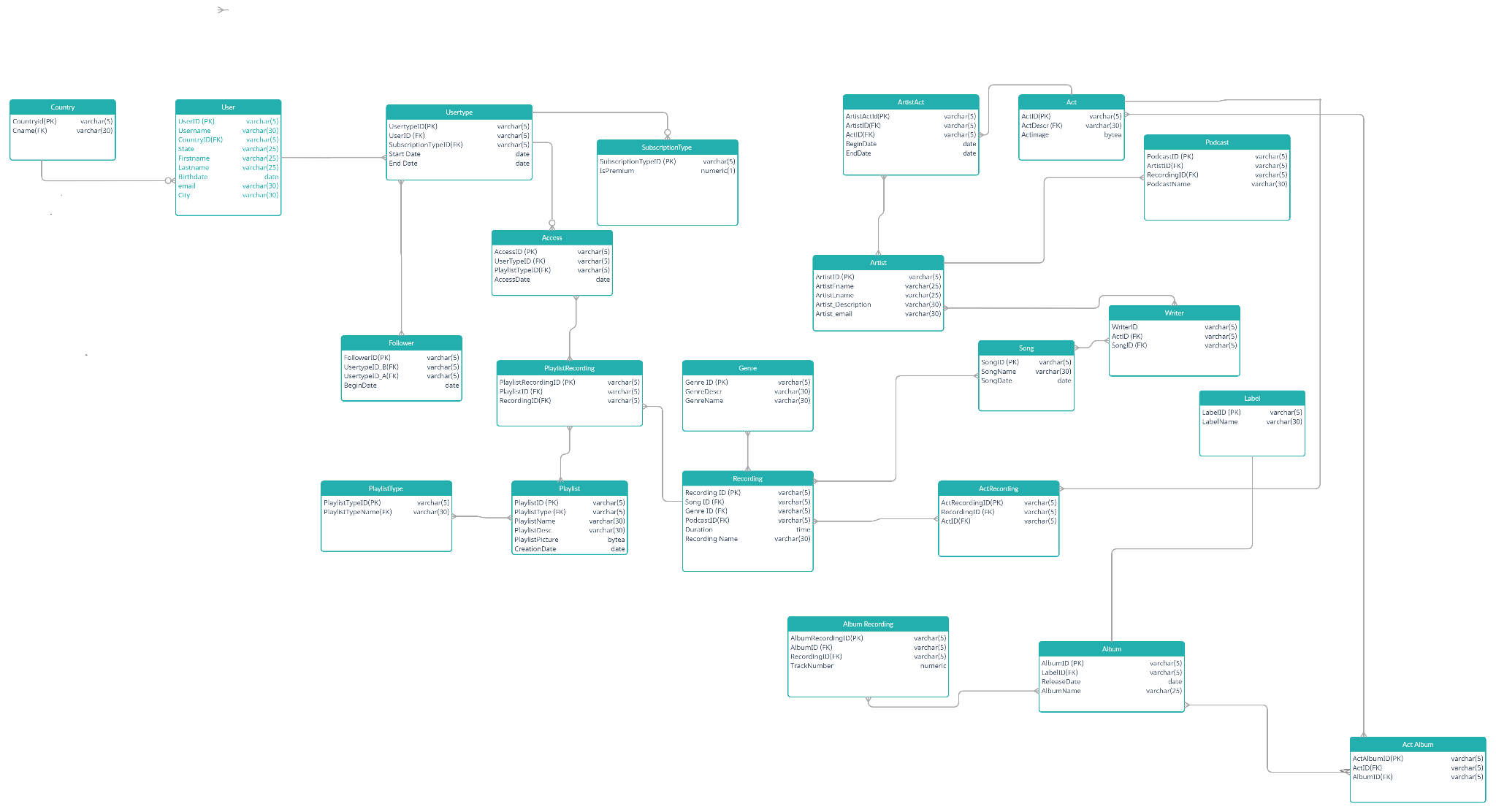
* Hertz is a music-playing application in which users can listen to their favourite songs as well as create personalized playlists as well as listen to the ones created by our developers.​
* Hertz looks forward to bringing podcasts to a new level as well as giving artists a smooth platform to record their songs and podcasts.​
* This app has a highly normalized database so as to make efficient access of data and reduce data redundancy.​

We tend to make this application more personalized.

**2. ENTITY-RELATIONSHIP MODEL**

**Link -** [**https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD**](https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD)

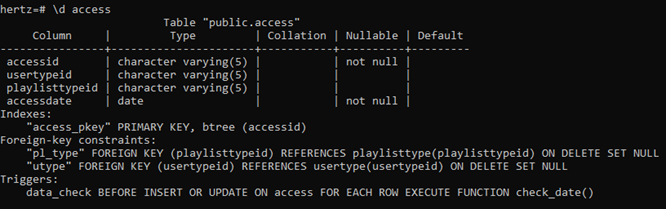
**3. RELATIONAL SCHEMA**



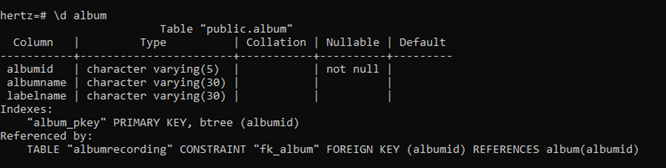
**Link -** [**https://app.creately.com/diagram/yQAR0D8Dgpa/edit**](https://app.creately.com/diagram/yQAR0D8Dgpa/edit)

**4.DATA DICTIONARY**

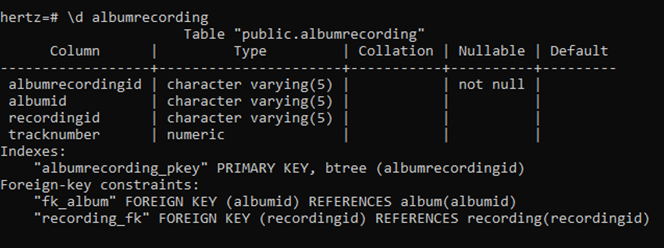
**4.1 access**



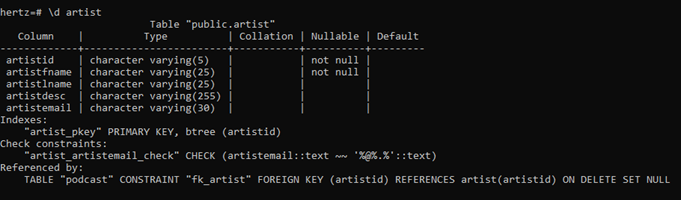
**4.2 album**

****

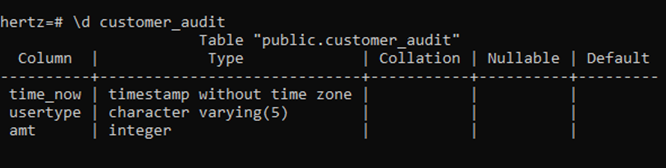
**4.3 album recording**



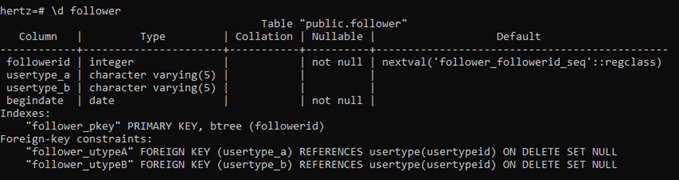
**4.4 artist**



**4.5 customer audit**



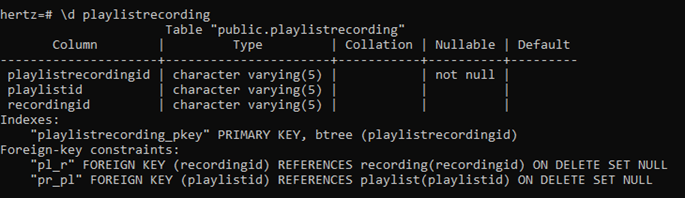
**4.6 follower**



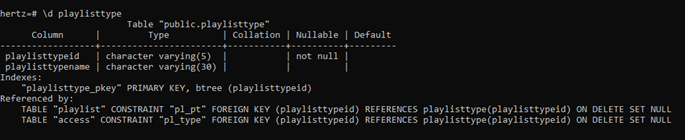
**4.7 playlist**



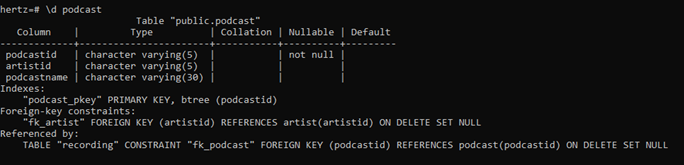
**4.8 playlist recording**



**4.9 playlisttype**



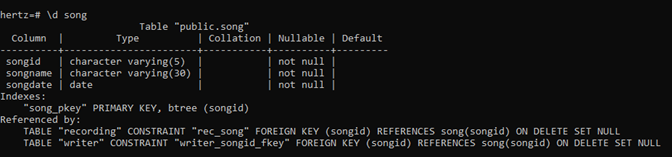
**4.10 podcast**



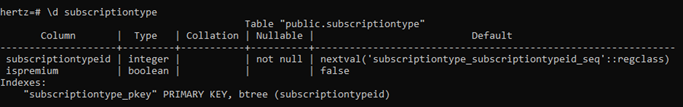
**4.11 recording**



**4.12 song**



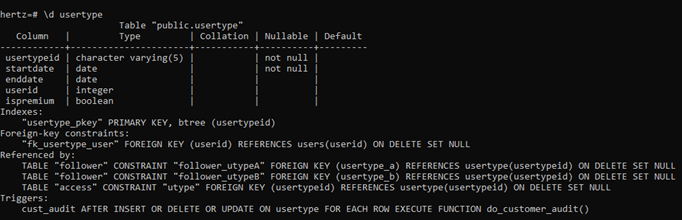
**4.13 subscription type**



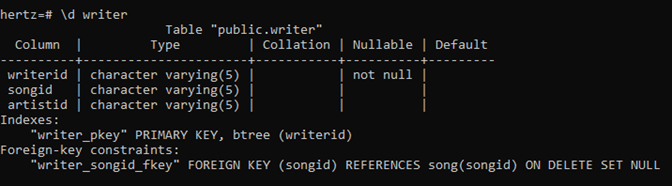
**4.14 users**



**4.15 usertype**



**4.16 writer**



**5. DATA IMPLEMENTATION**

**5.1 SCHEMA**

**5.1.1 access**

create table access (accessid varchar(5) primary key,usertypeid varchar(5) ,playlisttypeid varchar(5),accessdate date not null, constraint "pl\_type" foreign key(playlisttypeid) references playlisttype(playlisttypeid) on delete set null, constraint "utype" foreign key(usertypeid) references usertype(usertypeid) on delete set null);

**5.1.2 album**

create table album ( albumid varchar(5) primary key , albumname varchar(30),labelname varchar(30));

**5.1.3 albumrecording**

create table albumrecording(albumrecordingid varchar(5) primary key,albumid varchar(5),recordingid varchar(5),tracknumber numeric ,constraint "fk\_album" foreign key(albumid) references album(albumid),constraint "recording\_fk" foreign key(recordingid) references recording(recordingid));

**5.1.4 artist**

create table artist(artistid varchar(5) primary key,

artistfname varchar(25) not null,

artistlname varchar(25),

artistdesc varchar(255),

artistemail varchar(30));

alter table artist add constraint "artist\_artistemail\_check" check(artistemail like '%@%.%');

**5.1.5 customer\_audit**

create table customer\_audit( time\_now TIMESTAMP , usertype varchar(5) , amt INT);

**5.1.6 follower**

create table follower(

followerid serial primary key,

usertype\_A varchar(5),

usertype\_B varchar(5),

begindate date not null,

constraint "follower\_utypeA" foreign key (usertype\_A)

references usertype(usertypeid) on delete set null,

constraint "follower\_utypeB" foreign key (usertype\_B)

references usertype(usertypeid) on delete set null);

**5.1.7 playlist**

create table playlist(

playlistid varchar(5) primary key,

playlistname varchar(30) default 'default\_playlist',

playlistdesc varchar(255),

creationdate date not null,

playlisttypeid varchar(5) not null,

constraint "pl\_pt" foreign key (playlisttypeid)

references playlisttype(playlisttypeid) on delete set null);

**5.1.8 playlistrecording**

create table playlistrecording ( playlistrecordingid varchar(5) primary key ,playlistid varchar(5) , recordingid varchar(5) , constraint "pl\_r" foreign key (recordingid) references recording(recordingid) on delete set null, constraint "pr\_pl" foreign key (playlistid) references playlist(playlistid) on delete set null);

**5.1.9 playlisttype**

create table playlisttype(

playlisttypeid varchar(5) primary key,

playlisttypename varchar(30));

**5.1.10 podcast**

create table podcast(

podcastid varchar(5) primary key,

artistid varchar(5),

podcastname varchar(30),

constraint "fk\_artist" foreign key (artistid)

references artist(artistid) on delete set null);

**5.1.11 recording**

create table recording(

recordingid varchar(5) primary key,

recordingname varchar(25),

duration interval not null,

podcastid varchar(5),

songid varchar(5),

genre varchar(20),

constraint "fk\_podcast" foreign key (podcastid)

references podcast(podcastid) on delete set null,

constraint "rec\_song" foreign key (songid)

references song(songid) on delete set null);

**5.1.12 song**

create table song(

songid varchar(5) primary key,

songname varchar(30) not null,

songdate date not null);

**5.1.13 subscriptiontype**

create table subscriptiontype(

subscriptiontypeid serial primary key,

ispremium boolean default false);

**5.1.14 users**

create table users(

userid serial primary key,

username varchar(30) not null,

fname varchar(30) not null,

lname varchar(30),

birthdate date,

email varchar(50) not null,

city varchar(30),

state varchar(30),

country varchar(30),

unique(email),

unique(username));

alter table users add constraint "users\_email\_check" check(email like '%@%.%');

**5.1.15 usertype**

create table usertype(

usertypeid varchar(5) primary key,

startdate date not null,

enddate date,

userid integer,

ispremium boolean,

constraint "fk\_usertype\_user" foreign key (userid)

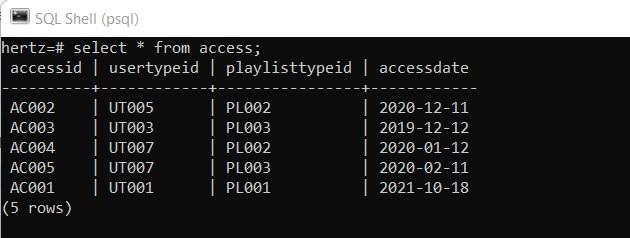
references users(userid) on delete set null);

**5.1.16 writer**

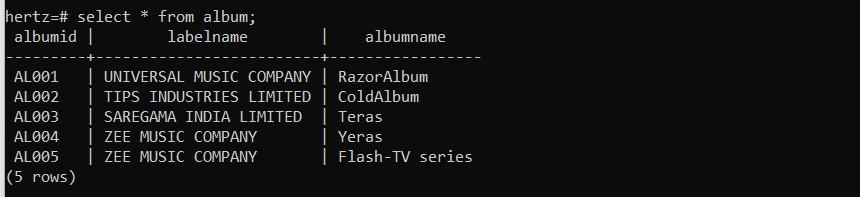
create table writer(writerid varchar(5) primary key not null , artistid varchar(5) , songid varchar(5) ,constraint "writer\_songid\_fkey" foreign key(songid) references song(songid) on delete set null);

**5.2 INSERTING DATA VALUES**

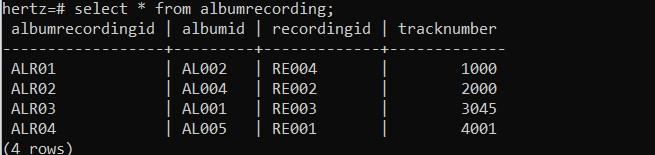
**5.2.1 Access:**

****

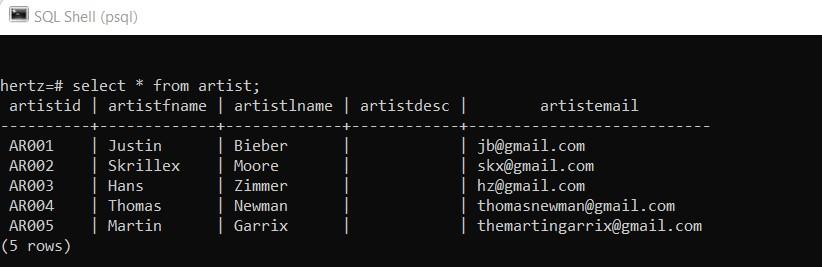
**5.2.2 Album :**

****

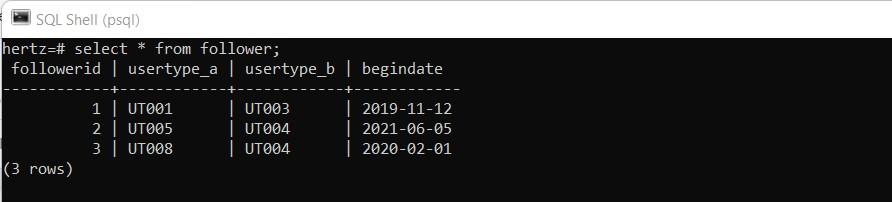
**5.2.3 Album Recording :**

****

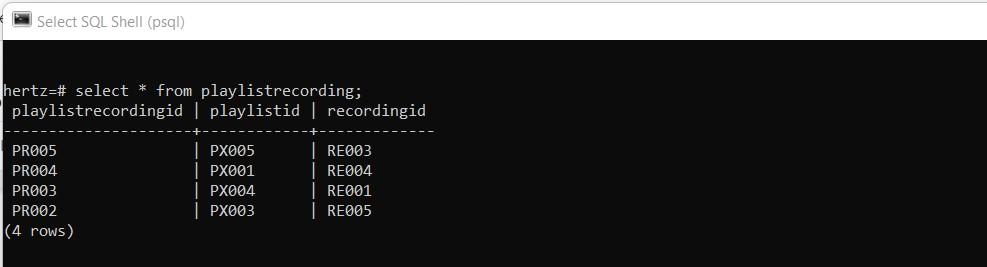
**5.2.4 Artist :**

****

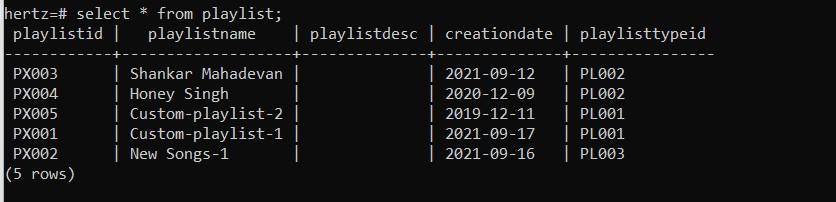
**5.2.5 Follower :**

****

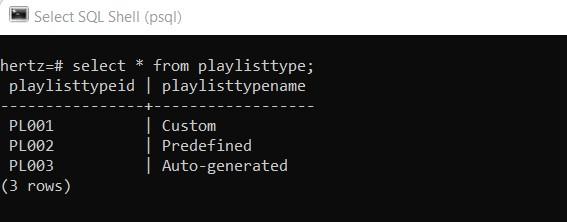
**5.2.6 Playlist Recording :**

****

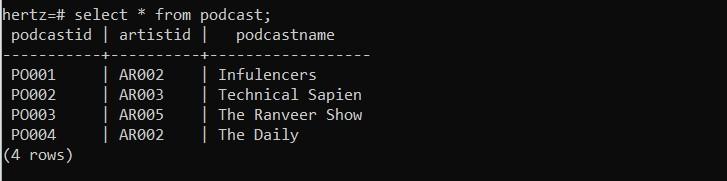
**5.2.7 Playlist :**

****

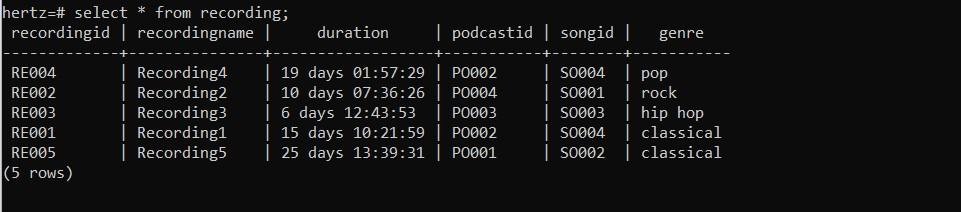
**5.2.8 Playlist Type :**

****

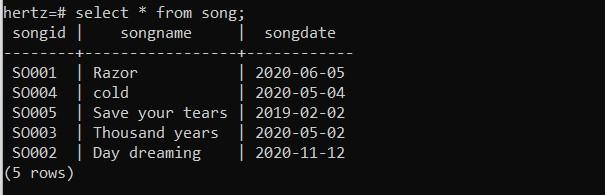
**5.2.9 Podcast :**

****

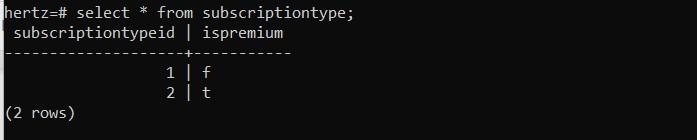
**5.2.10 Recording :**

****

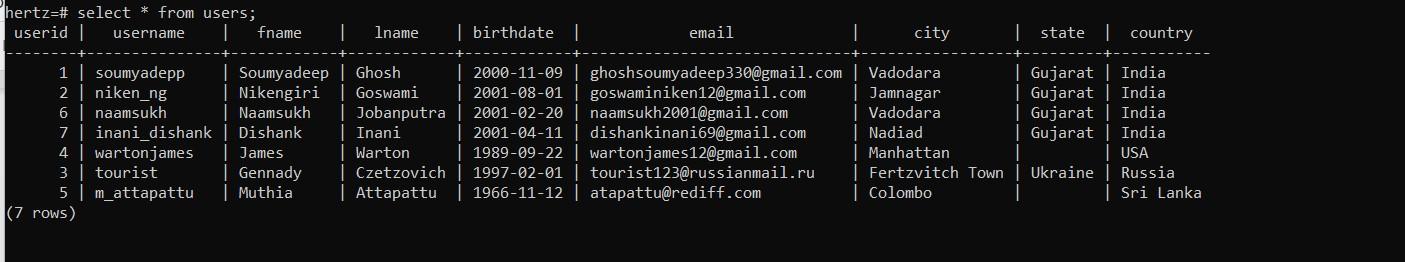
**5.2.11 Songs :**

****

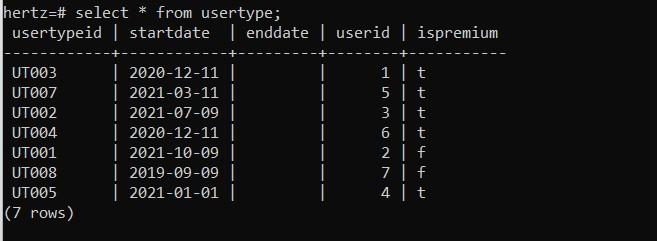
**5.2.12 SubscriptionType :**

****

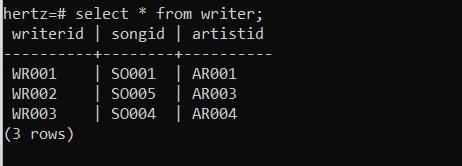
**5.2.13 Users :**

****

**5.2.14 User type :**

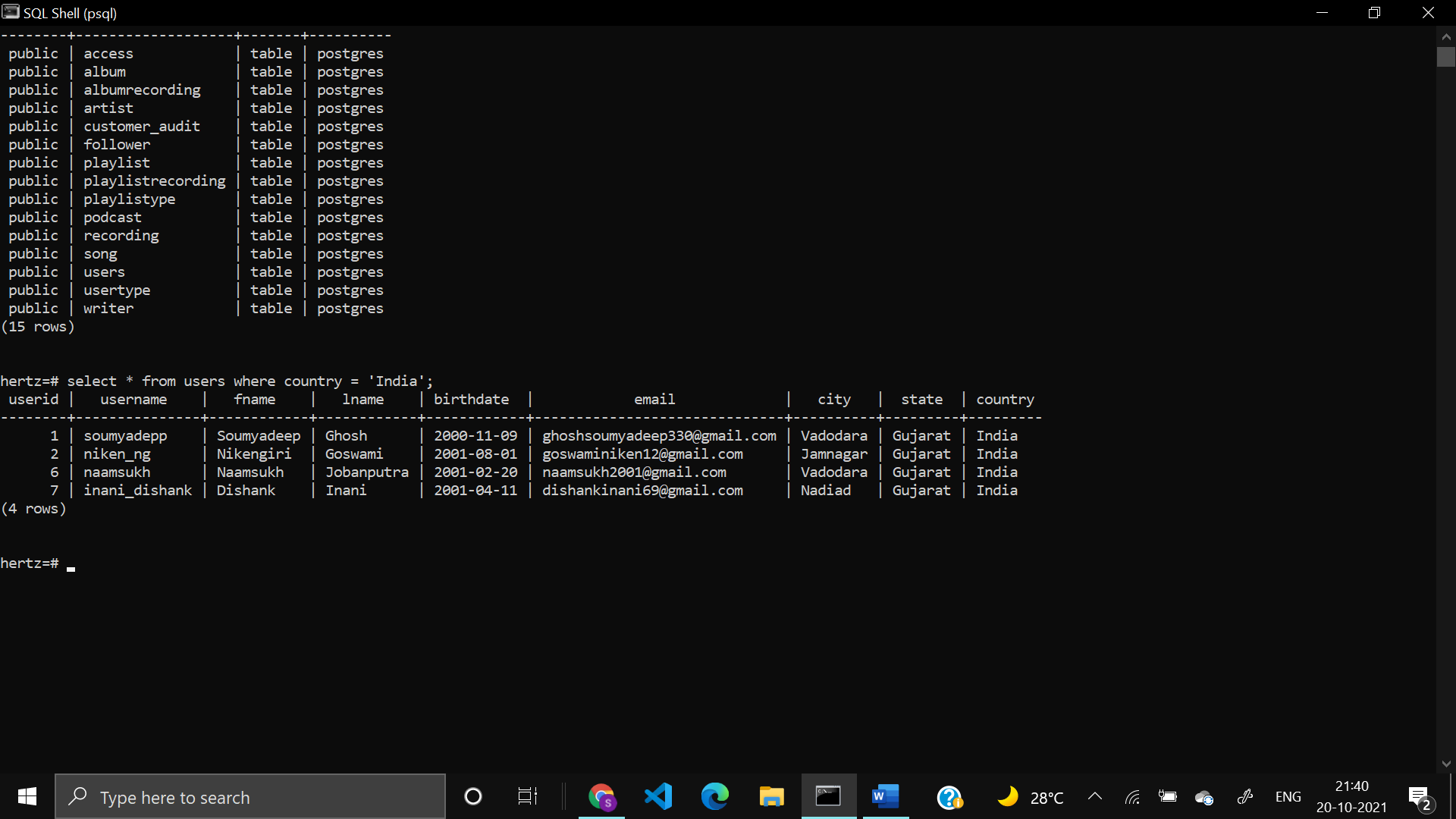
****

**5.2.15 Writer :**

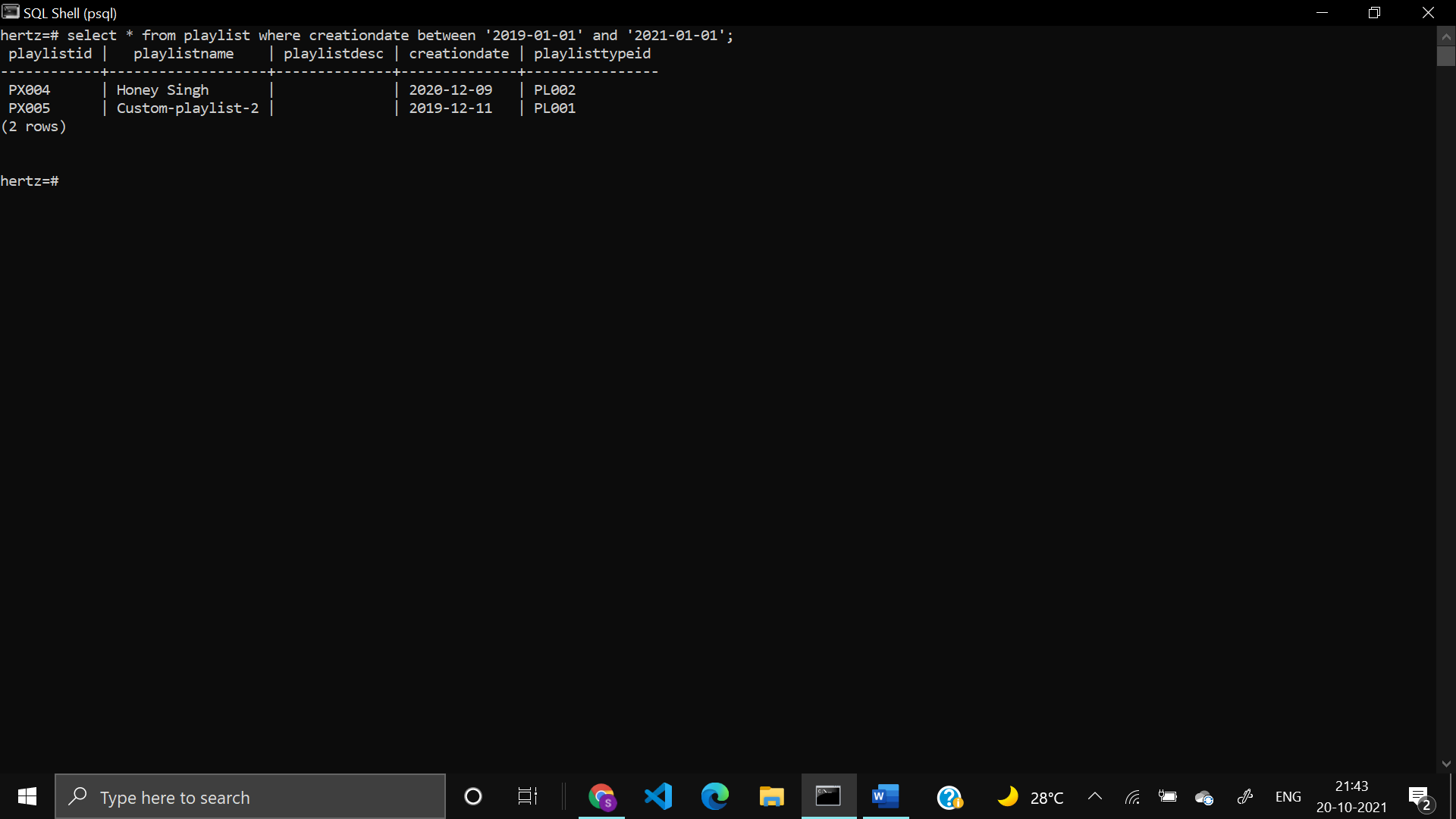


**5.3 QUERIES USING BASIC DBMS CONSTRUCTS:**

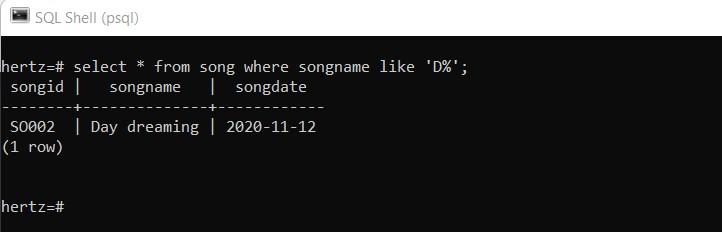
**5.3.1 Display the users who are from India.**

****

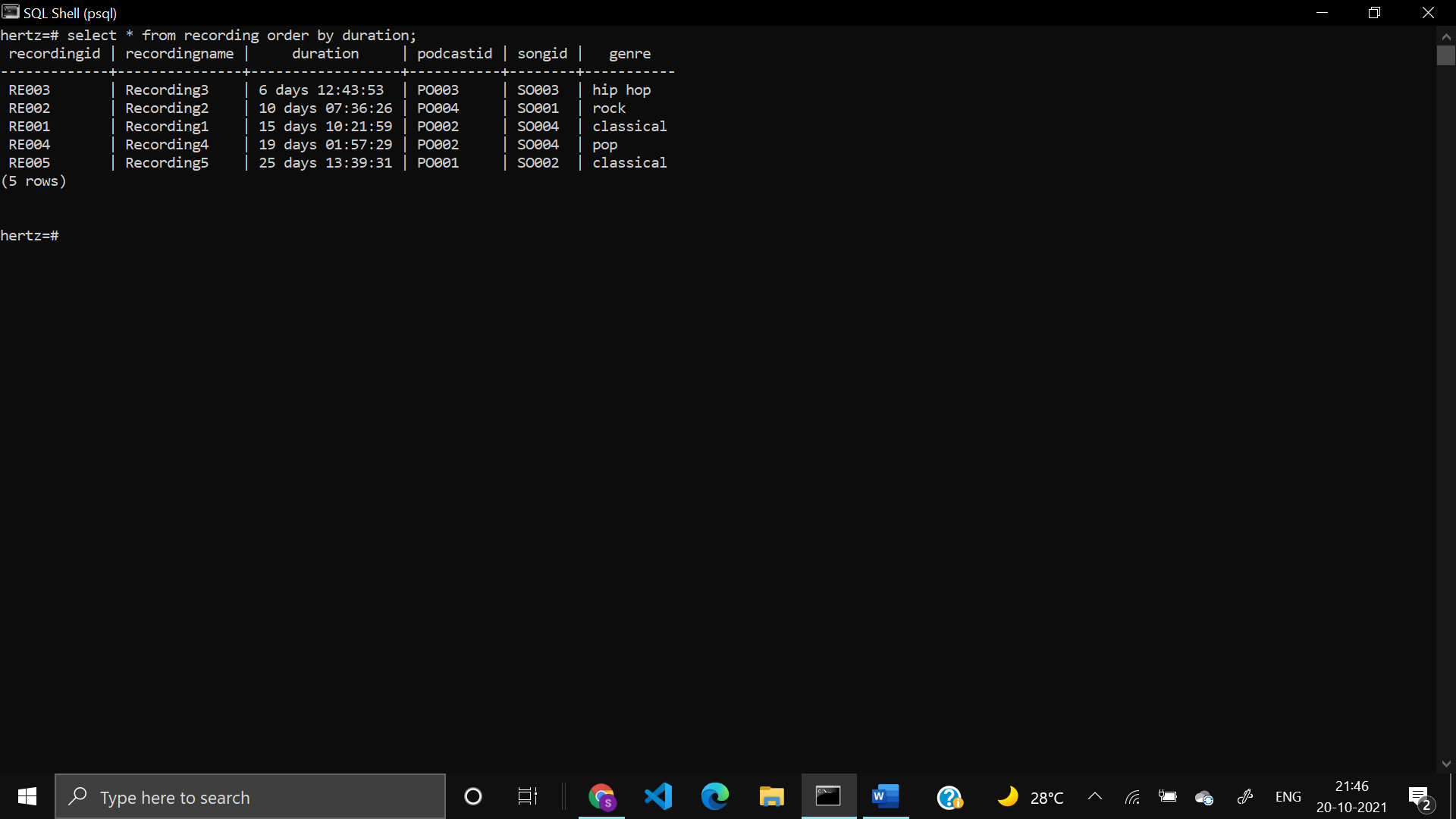
**5.3.2 Display the playlist which is created between year 2019-2021**

****

**5.3.3 Display the songs whose name start with D.**

****

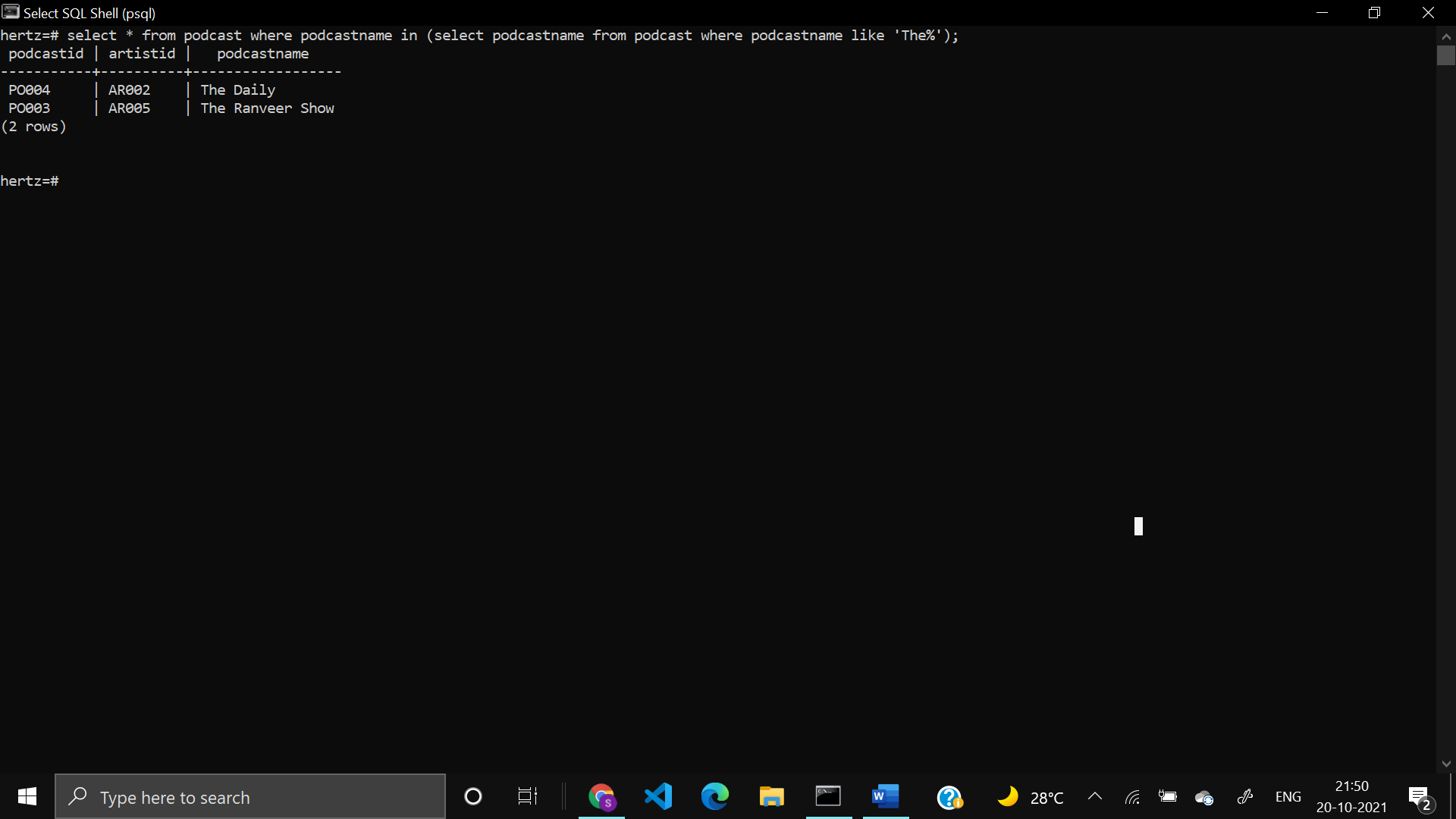
**5.3.4 Display the recording table and sort by duration.**

****

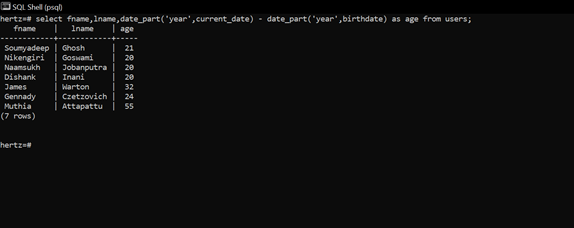
**5.3.5 Display the count of all the recordings by genre.**

****

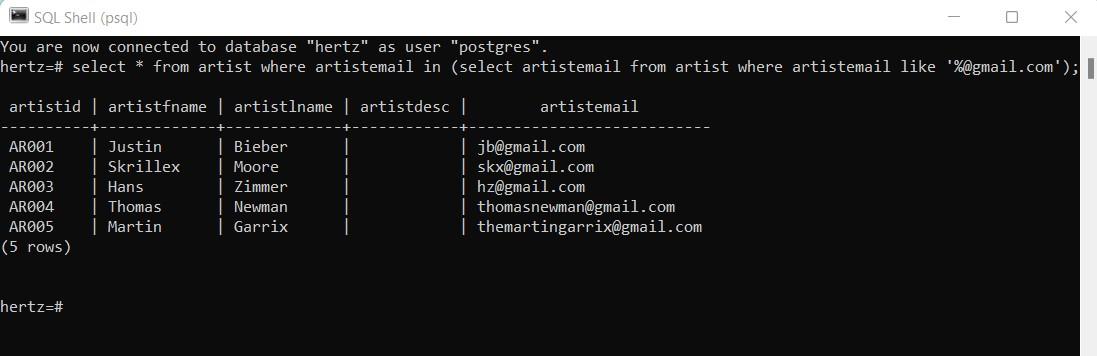
**5.3.6 Display the podcasts which starts with ‘THE’**

****

**5.3.7 Display the fname,lname of users with their age in ascending order.**

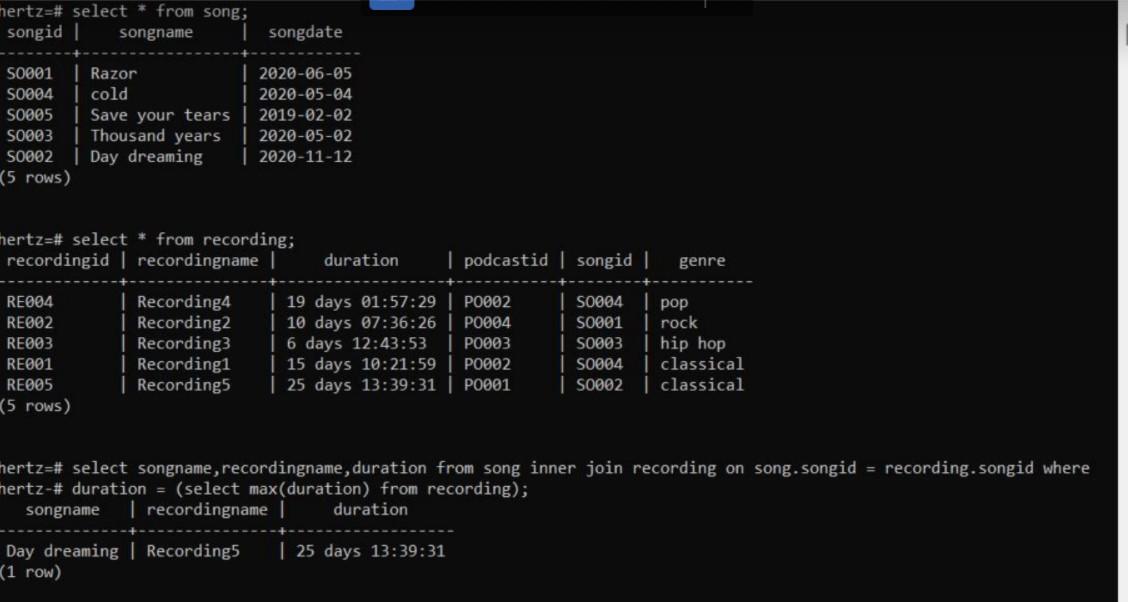
****

**5.3.8 Display all the artist who has a gmail account.**

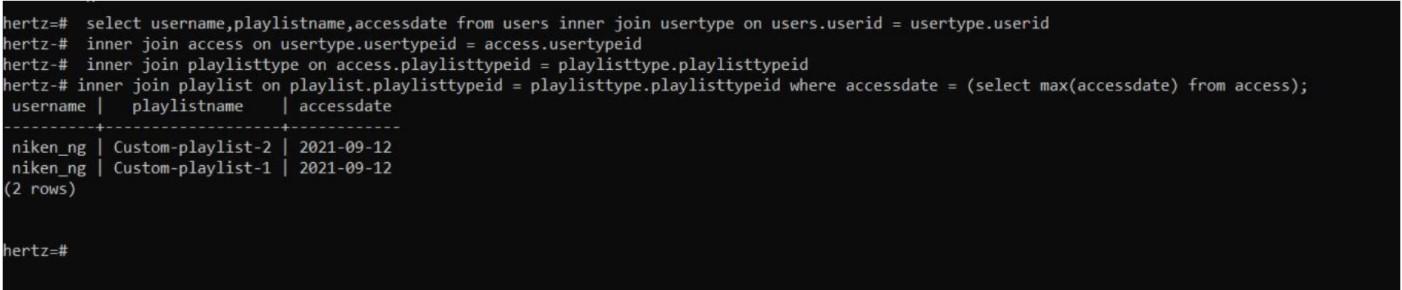
****

**5.4 QUERIES USING JOIN AND SUBQUERIES**

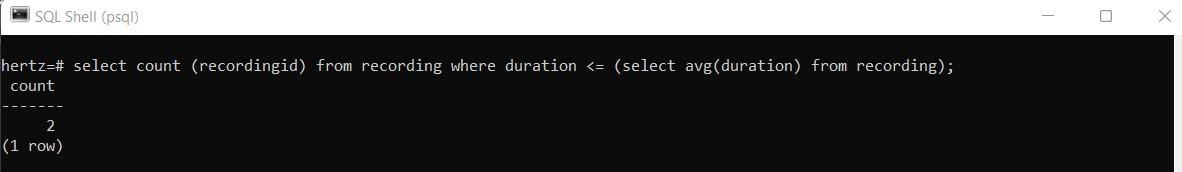
**5.4.1 Write a query to find the song name with maximum duration of recording**

****

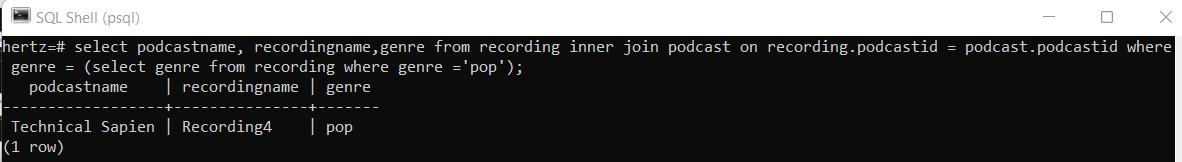
**5.4.2 Write a query which finds out the playlist which is been accessed recently by the users**

****

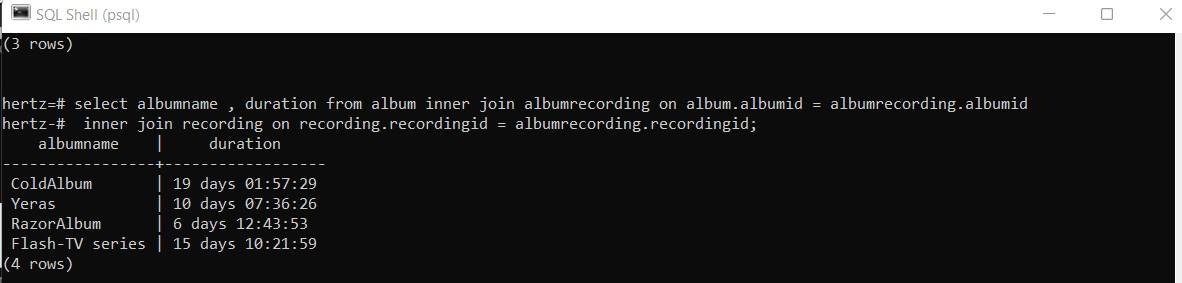
**5.4.3. Write a query that finds the recording which is having a duration less than equal to the average duration.**

****

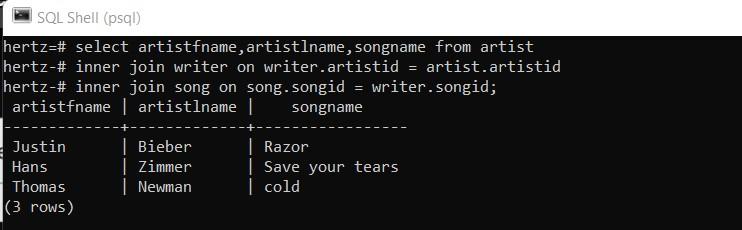
**5.4.4 Write a query which finds the podcast which is having a genre=’pop’.**

****

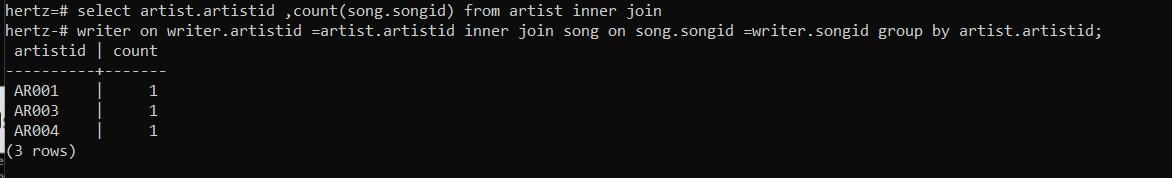
**5.4.5 Write a query which finds the album and its duration.**

****

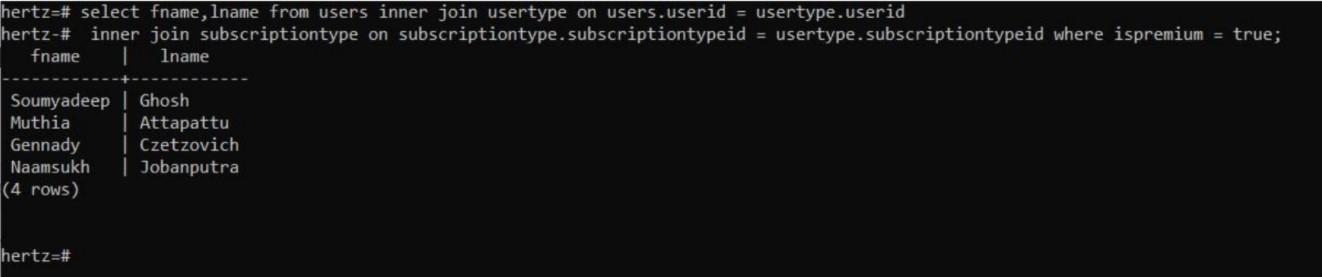
**5.4.6 Writer a query which finds the artist and the song written by him/her.**

****

**5.4.7 Write a query that finds the number of songs the artist has written.**

****

**5.4.8 Write a query that finds the users who have taken premium**

****

**5.5 PL/SQL:**

**VIEW**

Text

Description automatically generated

**ROWTYPE**

Text

Description automatically generated

* 1. **FUNCTION & TRIGGERS:**

**5.6.1 Function and trigger which keeps log of all the users who bought premium and make an entry in the customer\_audit table .**

**Function:**

CREATE OR REPLACE FUNCTION do\_customer\_audit() RETURNS TRIGGER

AS $customer\_audit$

BEGIN

IF NEW.ispremium = true THEN

INSERT INTO customer\_audit(time\_now,usertype,amt)

values(NOW(),NEW.usertypeid,199);

RETURN NEW;

END IF;

RETURN NULL;

END;

$customer\_audit$

LANGUAGE PLPGSQL;

**Trigger:**

CREATE TRIGGER cust\_audit

AFTER INSERT OR UPDATE OR DELETE on usertype

FOR EACH ROW EXECUTE PROCEDURE

do\_customer\_audit();

Text

Description automatically generated

**5.6.2 Function and trigger which checks user’s age and if it is less than 12 then it won’t allow to insert or update in user’s table.**

**FUNCTION :**

CREATE FUNCTION CHECK\_AGE() RETURNS TRIGGER AS $$

BEGIN

IF DATE\_PART('YEAR',CURRENT\_DATE) - DATE\_PART('YEAR',NEW.BIRTHDATE) < 12 THEN RAISE EXCEPTION 'AGE SHOULD BE ATLEAST 12';

END IF;

RETURN NEW;

END;

$$

LANGUAGE PLPGSQL;

**TRIGGER :**

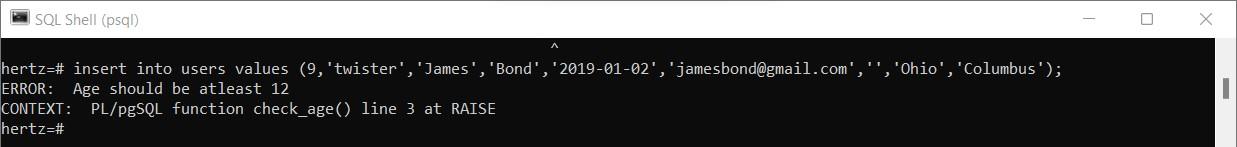
CREATE TRIGGER AGE\_CHECK

BEFORE INSERT OR UPDATE ON USERS

FOR EACH ROW EXECUTE PROCEDURE CHECK\_AGE();

Text

Description automatically generated

****

* 1. **CURSOR:**

**Create a Cursor which traverses through users table where the country of the user is INDIA.**

BEGIN

DECLARE mycursor CURSOR FOR

SELECT \* FROM USERS WHERE COUNTRY = 'India';

FETCH NEXT FROM mycursor;

FETCH PRIOR FROM mycursor;

CLOSE mycursor;

end;

**Text

Description automatically generated**

**6. FUTURE ENHANCEMENTS OF THE SYSTEM**

* We will design Front-end using React Framework and Develop Back-end in NodeJS
* Methods and user data input will be a lot easy after the implementation of GUI.
* In the future, we can place the system on the cloud so the maintenance of the data can be reduced.

**7. BIBLIOGRAPHY**

* We created ER-Model on Whimsical and Relational Schema on Creately
* ER-MODEL - [**https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD**](https://whimsical.com/YW63bK8pU6HZXs7F4h2YoD)
* RELATIONAL SCHEMA - [**https://app.creately.com/diagram/yQAR0D8Dgpa/edit**](https://app.creately.com/diagram/yQAR0D8Dgpa/edit)
* For the implementation of this project, we referred to materials shared by Prof. Archana N. Vyas and the following websites and books:

**Websites:**

* <https://www.w3schools.com/sql/sql_syntax.asp>
* <https://www.tutorialspoint.com/>
* <https://dev.mysql.com/doc/>
* <https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/>

**Book:**

Database System Concepts

-Henry F. Korth & A. Silberschatz 2nd Ed. McGraw-Hill 1991