DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES1201801987 - H M Thrupthi

Music Player System

Introduction:

Database system have become an essential component of everyday life in modern society and in that many frequently occurring activities involve the accessing of at least one database. Here it has been implemented through Music player system as mini world.
Some part of the real world information about which data is stored in a database. Here Music player system acts as mini world.

Music player system allows the user to store the details about a music artist, about his / her albums, even about the users who have sung those songs in the album. The system is strong enough to withstand regressive yearly operations under the conditions where database is maintained and cleared over certain time of span. The implementation of the above system will considerably reduce data entry, time and also provide readily calculated reports.

It keeps track of all the information about the artists in the music field, their position, status and total number of albums created by them. So simultaneously it keeps track of all the songs in an album and about the playlist associated with it. It also enables us to store details about the users who have sung those songs.

The system has been implemented with few constraints over the data to be stored in it. And also triggers been created to audit trail over every record been stored. The system also enables to do queries over the data stored easily, which would help to make the decisions over the artist or songs efficiently.

The Entity Relationship diagram represents the model of music player system.
The main Entities in the ER diagram are
Artist, Album, Song, Playlist, User_details.

- An artist can have many albums and an album can have many songs.
- Each playlist has list of songs which has been sung by particular user.
- The above system provides the information about artist as well as about the user who have sang the songs.

A trigger has been created over a table song, where if individual inserts a tuple to a table song, the time stamp, song id and duration have been listed in the other table called AUDIT.

Many queries have been done over it,

- List of all the artists who have won the national or padmashri awards.
- List of artist and corresponding albums
- Some Aggregated functional queries to calculate average, min, max of duration of songs.
- List the details of user who have played songs.
- Outer join queries
  1)List the playlist name, id, user name, for user email matching by left outer join.
  2) List the album name, id, artist name, for matching artist id by full outer join.

- The above system has a capability of storing information about the particular artist who has created the many albums. Can retrieve the useful information about the particular artist.
- It also has the information about awards that has been won by the artist, which helps us to know more about the artist current position.
- It also has details about the user who had played particular song, which helps know about the user.

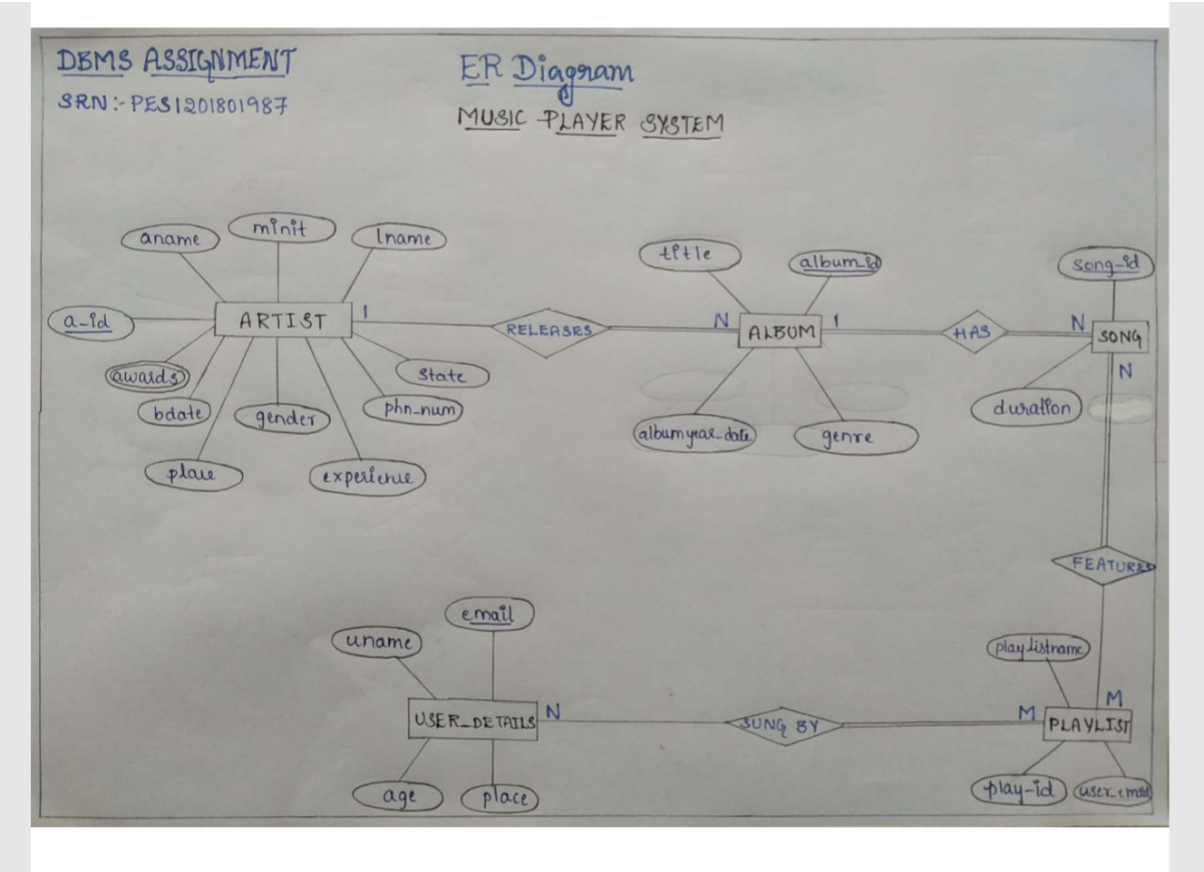
Introduction

In the music field many artists releases the albums, an album can have many songs in it. A particular playlist has the list of those songs and a particular user might have sung those songs who have their professionalism area in music field. The above system helps to store above information as form of database.
The main aim of the mini world is to store the information in a required manner about the music player system and their attributes, which would help in future to get know about the artist as well as his\her contribution in the field of music.

Data Model

A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey.

ER Diagram:



ER diagram represents model of the music player system, which shows all the virtual instrumental data tables in the system and relationship between the entities artist, album, song, playlist and user_details.
The following is the description of entities and its attributes:

1.ARTIST-

This table consists of details about the various artists from different places.The information stored in this table includes artist name, artist id, birth date, place, gender, experience and state in music field.
Constraint: artist name as aname will be unique for each artist and artist id as a_id a primary key.

2.ALBUM:

This table stores information about the albums created by artist. The table has the attributes like album title, album_id, album released year and date, genre and has artist id.
Constraint: Album title is unique for each album and the corresponding artist should exist in ARTIST table. Here art_id acts as foreign key referencing the table ARTIST.

3.SONG:

This table stores information about the songs in particular album.The table has the attributes like song_ title , song_id, duration of song and album id .
Constraint: song title is unique and the corresponding album
should exist in ALBUM table. Here al_id acts as foreign key referencing the table ALBUM.

4.PLAYLIST:

This table stores information about the playlist which as list of songs.The table has the attributes like playlist_name , play_id, user_email.
Constraint: Playlist name is unique and the corresponding list of songs has been stored in other table called played_songs. Play_id acts as primary key and user_email acts as foreign key referencing the table USER_DETAILS.

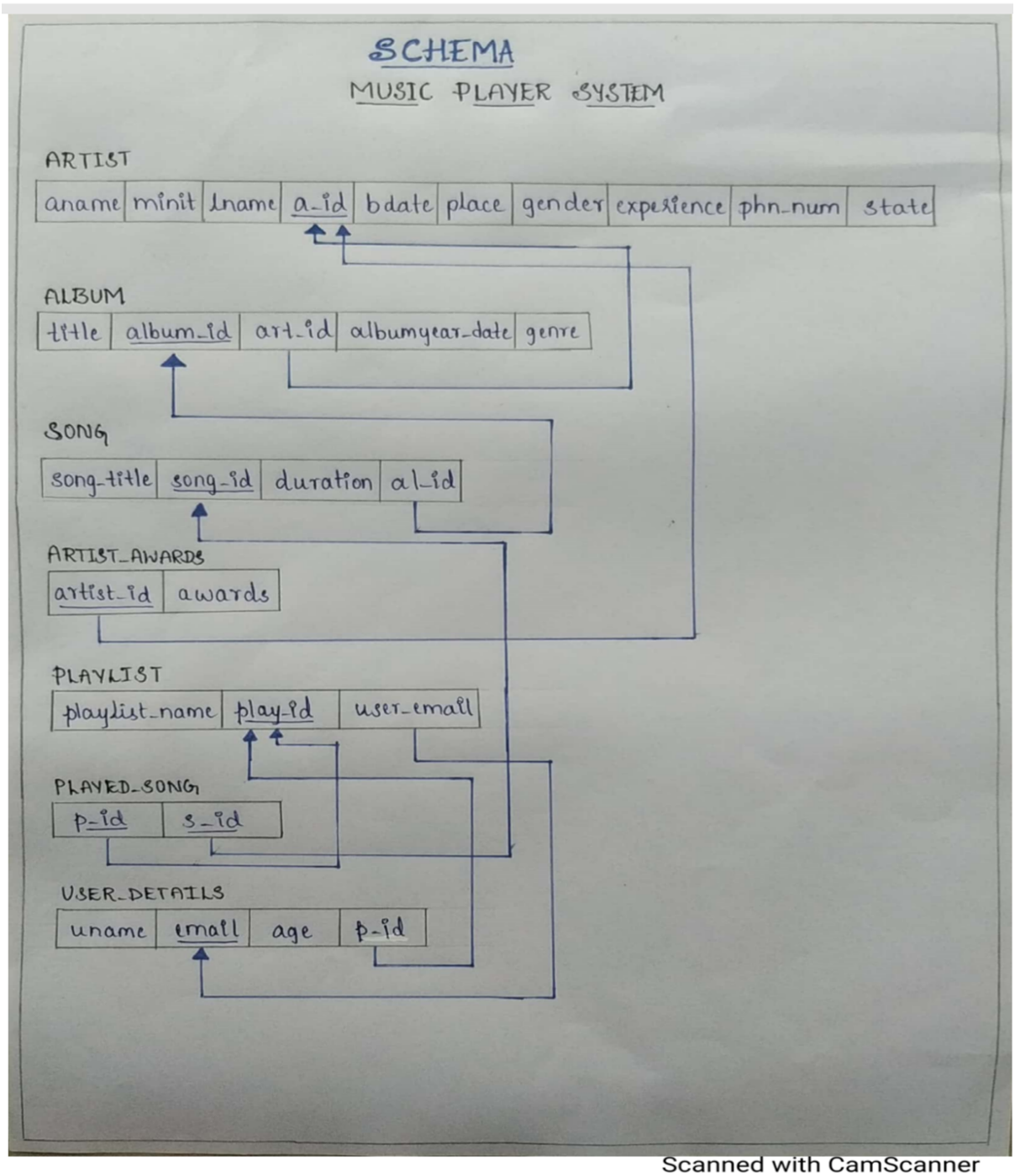
5.USER_DETAILS:

This table stores information about the user who has sung the particular playlist of songs. Which has user name as uname, email, age and p_id as corresponding playlist id.

Constraint: user name is unique and email acts as primary key and p_id acts as foreign key referencing the table PLAYLIST.

Over all integer, character and date data types has been used.

SCHEMA:

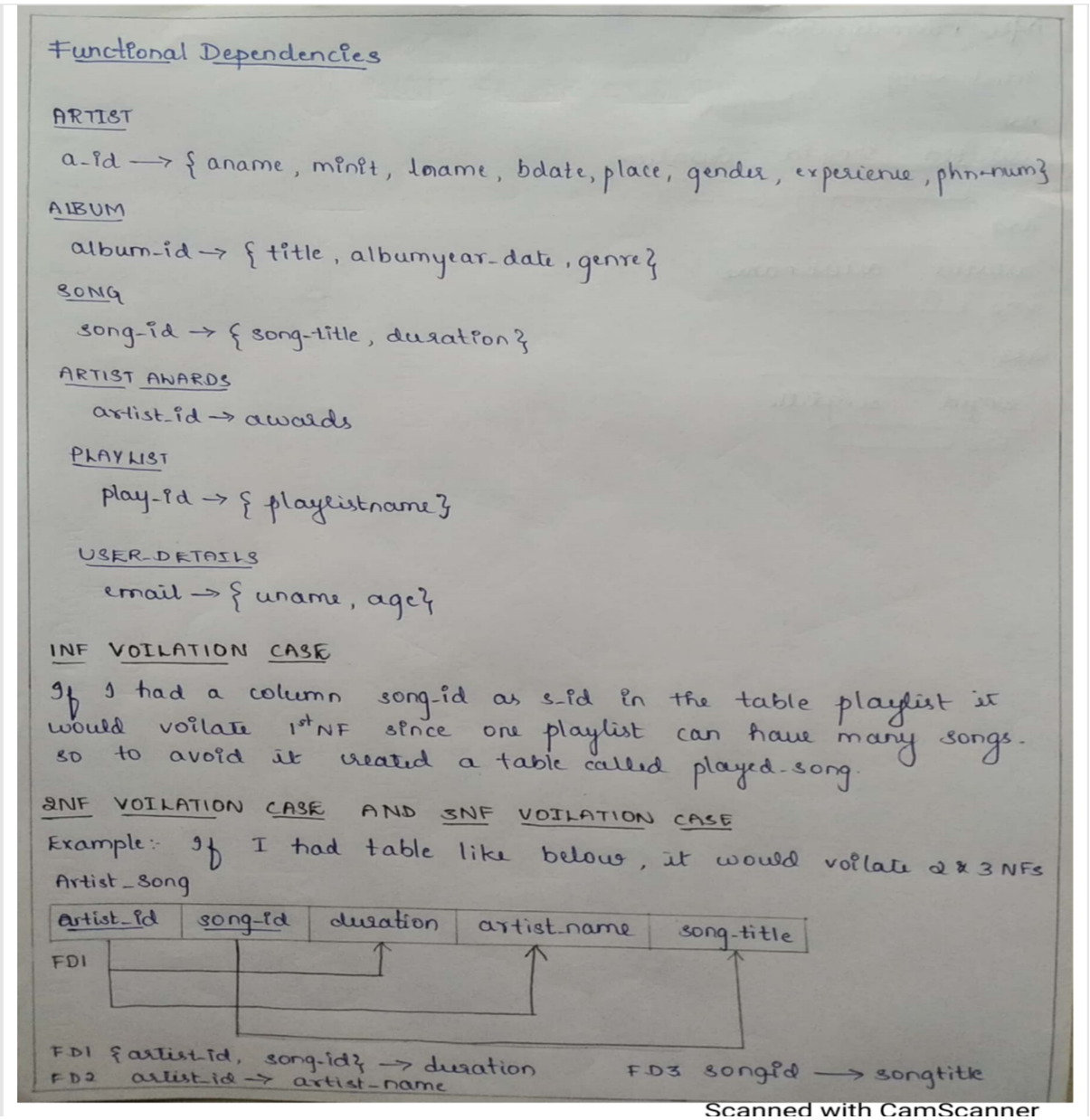


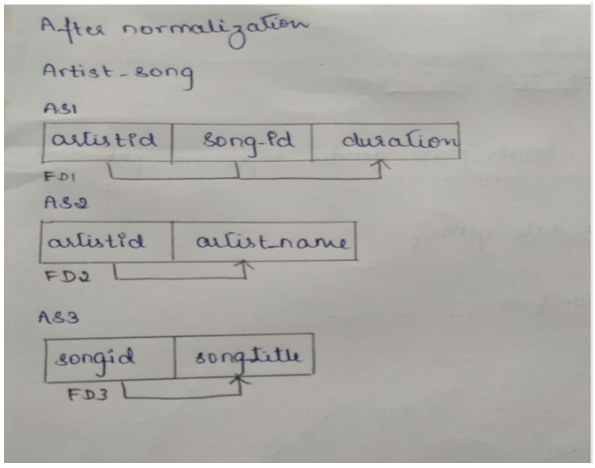
FD and Normalization

1st Normal Form:
In the table song, a particular song may sung for different duration of minutes by same user. So on that case its violates the first NF. In order to avoid that we can have a separate table as song_duration which lists the song and duration separately.

2nd Normal Form:
If I add a playlist name to the table play_song, 2nd NF gets violated because playid -> playlist_name but the key is play_id, s_id that is song id. Therefore 2nd normal form is violated if playlist name added in the play_song table.

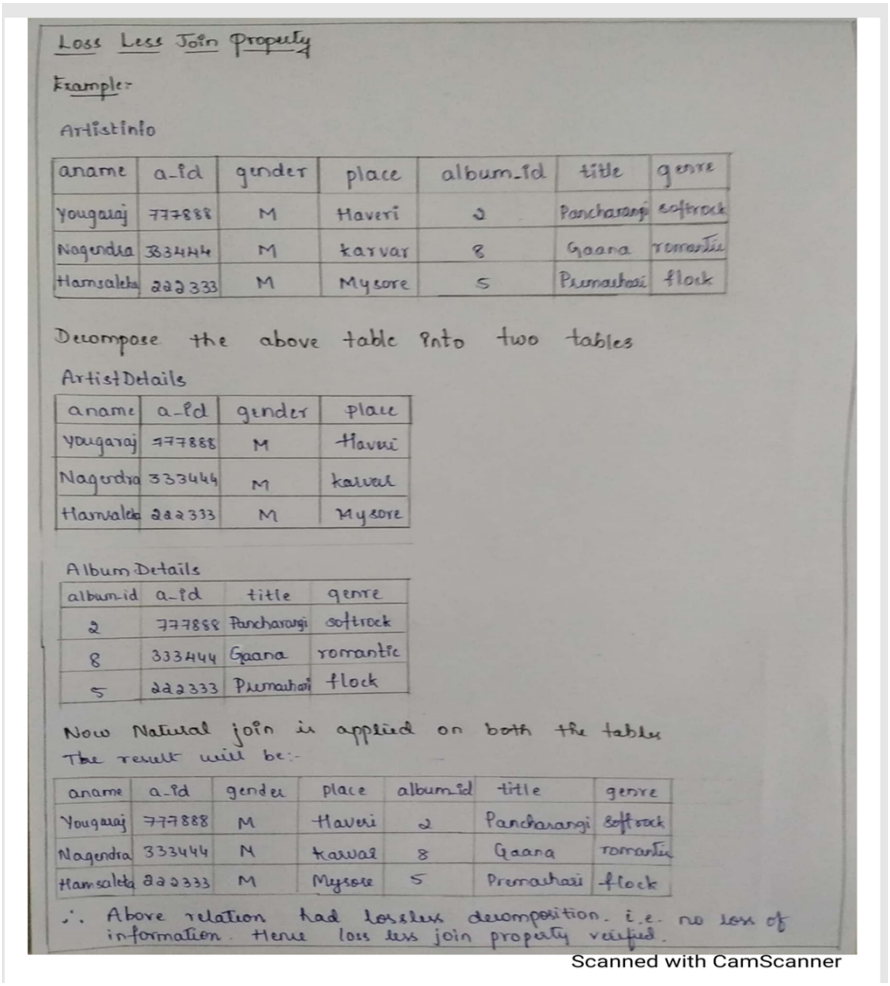
3rd Normal Form:
If I add an artist name along artist id into the table album, 3rd normal form gets violated. Same case happens if I add a album name along with album id into the table song.





Lossless join property verification:

Decomposition in DBMS removes redundancy, anomalies and inconsistencies from a database by dividing the table into multiple tables.
Decomposition is lossless if it is feasible to reconstruct relation R from decomposed tables using Joins. This is the preferred choice. The information will not lose from the relation when decomposed. The join would result in the same original relation. The example below represents it.



Loss Less Join Property

Example:-

ArtistInfo

aname	a_id	gender	place	album_id	title	genre
Yougaraj	777888	M	Haveri	2	Pancharangi	softrock
Nagendra	333444	M	karvar	8	Gaana	romantic
Hamsaleka	222333	M	Mysore	5	Premachari	flock

Decompose the above table into two tables

ArtistDetails

aname	a_id	gender	place
Yougaraj	777888	M	Haveri
Nagendra	333444	M	karvar
Hamsaleka	222333	M	Mysore

AlbumDetails

album_id	a_id	title	genre
2	777888	Pancharangi	softrock
8	333444	Gaana	romantic
5	222333	Premachari	flock

Now Natural join is applied on both the tables
The result will be:-

aname	a_id	gender	place	album_id	title	genre
Yougaraj	777888	M	Haveri	2	Pancharangi	softrock
Nagendra	333444	M	karvar	8	Gaana	romantic
Hamsaleka	222333	M	Mysore	5	Premachari	flock

∴ Above relation had lossless decomposition. i.e. no loss of information. Hence loss less join property verified.

DDL

Here is the below script describes how tables been created, datatypes been used and constraints over the tables.

```
--
-- PostgreSQL database dump
--

SET statement_timeout = 0;
SET lock_timeout = 0;
SET client_encoding = 'UTF8';
SET standard_conforming_strings = on;
SET check_function_bodies = false;
SET client_min_messages = warning;

--
-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:
--

CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg_catalog;


--
-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:
--

COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';


SET search_path = public, pg_catalog;

SET default_tablespace = '';

SET default_with_oids = false;

--
-- Name: album; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE album (
    title character varying(25) NOT NULL,
    album_id integer NOT NULL,
    art_id integer NOT NULL,
    albumyear_date date,
    genre character(15) NOT NULL
);
```

```
ALTER TABLE album OWNER TO postgres;

--
-- Name: song; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE song (
    song_id character(9) NOT NULL,
    song_title character varying(25) NOT NULL,
    duration numeric(5,1) NOT NULL,
    al_id integer NOT NULL

);


ALTER TABLE song OWNER TO postgres;

--
-- Name: artist_awards; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE artist_awards (
     artist_id integer NOT NULL,
     awards character varying(40) NOT NULL
);


ALTER TABLE artist_awards OWNER TO postgres;

--
-- Name: artist; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE artist (
    aname character varying(35) NOT NULL,
    minit character(1),
    lname character varying(15) NOT NULL,
    a_id integer NOT NULL,
    bdate date,
    place character varying(10),
    gender character(1),
    experience numeric(10,2),
    phn_num character(10),
    state character varying(20)
);
```

```
ALTER TABLE artist OWNER TO postgres;

--
-- Name: played; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE playlist (
    playlistname character varying(15) NOT NULL,
    play_id integer NOT NULL,
    user_email character varying(40)
);


ALTER TABLE  playlist OWNER TO postgres;

--
-- Name: played_song; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE played_song (
    p_id integer NOT NULL,
    s_id character(9) NOT NULL

);


ALTER TABLE played_song OWNER TO postgres;

--
-- Name: user_details; Type: TABLE; Schema: public; Owner: postgres; Tablespace:
--

CREATE TABLE user_details (
    uname character varying(45) NOT NULL,
    email character varying(25) NOT NULL,
    age integer NOT NULL,
    p_id integer NOT NULL);
ALTER TABLE user_details OWNER TO postgres;

--
-- Data for Name: album; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY album (album_id, title, art_id, albumyear_date, genre) FROM stdin;
1       Thriller 111222          1986-04-05     pop
2       Pancharangi   777888          1958-05-03     softrock
4       Sathyagragh   444555          2002-06-18     jazz
5       Premachari     222333          1997-08-26     flock
```

```
6	songofyouth	666777	2012-04-30	petriotic
7	bodyguard	666777	1980-07-26	pop
8	gaana	333444	2000-05-25	romantic
9	Premalokha	222333	1988-09-26	romantic
10	Agumbe	222333	1970-09-27	romantic
\.

--
-- Data for Name: artist_awards; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY artist_awards (artist_id, awards) FROM stdin;
222333	NationalAward
333444	KarnatakaStateAward
777888	BestLyricist
111222	Padmashri
222333	StatefilmfareAward
111222	Barathratna
666777	NationalAward
\.

--
-- Data for Name: artist; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY artist (aname, minit, lname, a_id, bdate, place, gender, experience, phn_num, state)
FROM stdin;
MichaelJackson	E	Borg	111222	1937-11-10	chicago	M	35	9591487525	Bihar
Hamsalekha	B	Wong	222333	1951-06-23	Mysuru	M	45	8415685624	karnataka
Yuvashankan	K	Raj	444555	1971-11-10	chenai	M	37	9024512755	Tamilnadu
Nagendra	P	Prasd	333444	1965-01-22	Mandya	M	36	7465378244	karnataka
Yougaraj	B	Bhat	777888	1965-01-09	Haveri	M	40	9876590487	karnataka
Jayanthi	G	Rao	555666	1954-06-22	karvar	F	43	9843221388	karnataka
Baskaran	R	Bhin	666777	1947-08-15	trivendram	M	47	9809332456	Kerala
\.


--
-- Data for Name: song; Type: TABLE DATA; Schema: public; Owner: postgres
--
```

```
COPY song (song_title, song_id, duration, al_id) FROM stdin;
Yaripremachari	A	5.8	5
Rannano	B	4.5	5
Hoovantha	C	5.20	5
Jaisriram	D	6.4	8
Dosthakano	E	4.58	8
Robertheme	F	5.5	8
Papamere	G	6.2	4
Terimeri	H	4.2	4
Ghoomar	I	3.6	2
Lifuistene	J	4.6	2
Udisuve	K	5.0	2
\.

--
-- Data for Name:  playlist; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY  playlist (playlistname, play_id, user_email) FROM stdin;
Hamsalekhaplay	11	rajesh@gmail.com
Nagendraplay 22	vijayprakash@gmail.com
Yuvashankan  33	shreyagoshal@gmail.com
Yougarajplay  44	chaithra@gmail.com
\.

--
-- Data for Name: played_song; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY played_song (p_id, s_id) FROM stdin;
11	A
11	B
11	C
22	D
22	E
22	F
33	G
33	H
33	I
44	J
44	K
\.

--
-- Data for Name: user_details; Type: TABLE DATA; Schema: public; Owner: postgres
--

COPY user_details (uname, email, age, p_id) FROM stdin;
```

```
RajeshKrishnan	rajesh@gmail.com	55	11
Vijayprakash	vijayprakash@gmail.com	50	22
Shreyagoshal	shreyagoshal@gmail.com	35	33
Chaithra	chaithra@gmail.com	34	44
\.

--
-- Name: album_title_key; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY album
    ADD CONSTRAINT album_title_key UNIQUE (title);

--
-- Name: album_album_idkey; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY album
    ADD CONSTRAINT album_album_idkey PRIMARY KEY (album_id);


--
-- Name: song_id_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY song
    ADD CONSTRAINT song_id_pkey PRIMARY KEY (song_id);

--
-- Name: song_title_key; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--
ALTER TABLE ONLY song
ADD CONSTRAINT song_title_key UNIQUE (song_title);
--
-- Name: artist_awards_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY artist_awards
    ADD CONSTRAINT artist_awards_pkey PRIMARY KEY (artist_id, awards);


--
-- Name: played_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
```

```
--

ALTER TABLE ONLY  playlist
    ADD CONSTRAINT played_pkey PRIMARY KEY (play_id);


--
-- Name: artist_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres; Tablespace:
--

ALTER TABLE ONLY artist
    ADD CONSTRAINT artist_pkey PRIMARY KEY (a_id);


--
-- Name: artist_aname_key; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY artist
    ADD CONSTRAINT artist_aname_key UNIQUE (aname);

--
-- Name: user_details_uname_key; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY user_details
    ADD CONSTRAINT user_details_email_pkey PRIMARY KEY (email);
--
-- Name: user_details_email_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY user_details
    ADD CONSTRAINT user_details_uname_key UNIQUE (uname);
--
-- Name: played_song_pkey; Type: CONSTRAINT; Schema: public; Owner: postgres;
Tablespace:
--

ALTER TABLE ONLY played_song
    ADD CONSTRAINT played_song_pkey PRIMARY KEY (p_id, s_id);

--
-- Name: album_art_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY album
    ADD CONSTRAINT  album_art_id_fkey FOREIGN KEY (art_id) REFERENCES
artist(a_id);
--
-- Name: song_al_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--

ALTER TABLE ONLY song
    ADD CONSTRAINT song_al_id_fkey FOREIGN KEY (al_id) REFERENCES
album(album_id);



--
-- Name: played_song_p_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY played_song
    ADD CONSTRAINT played_song_p_id_fkey FOREIGN KEY (p_id) REFERENCES
playlist(play_id);




--
-- Name: artist_awards_artist_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY artist_awards
    ADD CONSTRAINT artist_awards_artist_id_fkey FOREIGN KEY (artist_id)
REFERENCES artist(a_id);




--
-- Name: played_user_email_fkey; Type: FK CONSTRAINT; Schema: public; Owner:
postgres
--

ALTER TABLE ONLY  playlist
    ADD CONSTRAINT played_user_email_fkey FOREIGN KEY (user_email)
REFERENCES user_details(email);



--
-- Name: user_p_id_fkey; Type: FK CONSTRAINT; Schema: public; Owner: postgres
--
```

```
ALTER TABLE ONLY user_details
    ADD CONSTRAINT user_p_id_fkey FOREIGN KEY (p_id) REFERENCES
playlist(play_id);


--
-- Name: public; Type: ACL; Schema: -; Owner: postgres
--

REVOKE ALL ON SCHEMA public FROM PUBLIC;
REVOKE ALL ON SCHEMA public FROM postgres;
GRANT ALL ON SCHEMA public TO postgres;
GRANT ALL ON SCHEMA public TO PUBLIC;


--
-- PostgreSQL database dump complete
--
```

Triggers

Here want to keep audit trial for every record being inserted in SONG table, to keep audit trial, we will create a new table called AUDIT where log messages will be inserted whenever there is an entry in SONG table for a new record.

Here, ID is the AUDIT record ID, and s_ID is the ID, duration of song as s_duration which will come from SONG table, and DATE will keep timestamp when the record will be created in SONG table. The following will create a trigger on SONG table as follows –

```
CREATE TABLE AUDIT(
   s_ID character(9) NOT NULL,
   s_duration numeric(5,1) NOT NULL,
   ENTRY_DATE TEXT NOT NULL);
CREATE OR REPLACE FUNCTION auditlogfunc() RETURNS TRIGGER AS
$example_table$

   BEGIN
      INSERT INTO AUDIT(s_ID, s_duration, ENTRY_DATE) VALUES (new.song_id,
new.duration, current_timestamp);
      RETURN NEW;
   END;
$example_table$ LANGUAGE plpgsql;

CREATE TRIGGER example_trigger AFTER INSERT ON song
FOR EACH ROW EXECUTE PROCEDURE auditlogfunc();
```

```
nb=# \i C:/Users/hp/Desktop/trigger.sql
CREATE TABLE
CREATE FUNCTION
CREATE TRIGGER
nb=# INSERT INTO song (song_title, song_id, duration, al_id) VALUES('ng','k','1.2','5');
INSERT 0 1
nb=# select * from AUDIT;
   s_id    | s_duration |            entry_date
-----------+------------+----------------------------------
 k         |        1.2 | 2020-05-30 12:07:29.961611+05:30
(1 row)


nb=#
```

SQL Queries

Simple queries:

List the details of artist and album of artist who have released the album.
select * from album, artist where art_id=a_id;

```
ghj=# select * from  album, artist where art_id=a_id;
   title    | album_id | art_id | albumyear_date |     genre     |     aname     | minit | lname | a_id |   bdate    |   place   | gender | experience |  phn_num   |   state
------------+----------+--------+----------------+---------------+---------------+-------+-------+------+------------+-----------+--------+------------+------------+-----------
 Thriller   |        1 | 111222 | 1986-04-05     | pop           | MichaelJackson | E     | Borg  | 111222 | 1937-11-10 | chicago   | M      |      35.00 | 9591487525 | Bihar
 Pancharangi |        2 | 777888 | 1958-05-03     | softrock      | Yougaraj      | B     | Bhat  | 777888 | 1965-01-09 | Haveri    | M      |      40.00 | 9876590487 | karnataka
 Sathyagragh |        4 | 444555 | 2002-06-18     | jazz          | Yuvashankan   | K     | Raj   | 444555 | 1971-11-10 | chenai    | M      |      37.00 | 9024512755 | Tamilnadu
 Premachari |        5 | 222333 | 1997-08-26     | flock         | Hamsalekha    | B     | Wong  | 222333 | 1951-06-23 | Mysuru    | M      |      45.00 | 8415685624 | karnataka
 songofyouth |        6 | 666777 | 2012-04-30     | petriotic     | Baskaran      | R     | Bhin  | 666777 | 1947-08-15 | trivendram | M      |      47.00 | 9809332456 | Kerala
 bodyguard  |        7 | 666777 | 1980-07-26     | pop           | Baskaran      | R     | Bhin  | 666777 | 1947-08-15 | trivendram | M      |      47.00 | 9809332456 | Kerala
 gaana      |        8 | 333444 | 2000-05-25     | romantic      | Nagendra      | P     | Prasd | 333444 | 1965-01-22 | Mandya    | M      |      36.00 | 7465378244 | karnataka
 Premalokha |        9 | 222333 | 1988-09-26     | romantic      | Hamsalekha    | B     | Wong  | 222333 | 1951-06-23 | Mysuru    | M      |      45.00 | 8415685624 | karnataka
 Agumbe     |       10 | 222333 | 1970-09-27     | romantic      | Hamsalekha    | B     | Wong  | 222333 | 1951-06-23 | Mysuru    | M      |      45.00 | 8415685624 | karnataka
(9 rows)
```

List the song details .

select song_title, duration, song_id from song;

```
ghj=# select song_title, duration, song_id from song;
   song_title    | duration |  song_id
-----------------+----------+-----------
 Yaripremachari |      5.8 | A
 Rannano        |      4.5 | B
 Hoovantha      |      5.2 | C
 Jaisriram      |      6.4 | D
 Dosthakano     |      4.6 | E
 Robertheme     |      5.5 | F
 Papamere       |      6.2 | G
 Terimeri       |      4.2 | H
 Ghoomar        |      3.6 | I
 Lifuistene     |      4.6 | J
 Udisuve        |      5.0 | K
(11 rows)


ghj=#
```

Nested queries:

1)To select artist details who have won national and padmashri award.

```
select aname, minit, lname, place, experience, phn_num
from artist
where a_id in ( select artist_id from artist_awards where awards in ('NationalAward',
'Padmashri'));
```

```
ghj=# select aname, minit, lname, place, experience, phn_num from artist where a_id in ( select artist_id from artist_awards where awards in ('NationalAward', 'Padmashri'));
    aname     | minit | lname |  place  | experience |  phn_num
--------------+-------+-------+---------+------------+-----------
 MichaelJackson | E     | Borg  | chicago |      35.00 | 9591487525
 Hamsalekha     | B     | Wong  | Mysuru  |      45.00 | 8415685624
(2 rows)


ghj=#
ghj=#
```

2)List the details of artist who have created albums (Using EXISTS)

```
select aname, minit, lname, bdate, place
from artist a
 where exists (select art_id from album al where a.a_id = al.art_id);
```

```
ghj=#
ghj=#  select aname, minit, lname, bdate, place from artist a where exists (select art_id from album al where a.a_id = al.art_id);
    aname     | minit | lname |   bdate    |   place
--------------+-------+-------+------------+-----------
 MichaelJackson | E     | Borg  | 1937-11-10 | chicago
 Hamsalekha     | B     | Wong  | 1951-06-23 | Mysuru
 Yuvashankan    | K     | Raj   | 1971-11-10 | chenai
 Nagendra       | P     | Prasd | 1965-01-22 | Mandya
 Yougaraj       | B     | Bhat  | 1965-01-09 | Haveri
 Baskaran       | R     | Bhin  | 1947-08-15 | trivendram
(6 rows)
```

3)List the details of artist who doesn't have created albums (Using NOT EXISTS)

```
select aname, minit, lname, bdate, place
 from artist a
where not exists (select art_id from album al where a.a_id = al.art_id);
```

```
ghj=# select aname, minit, lname, bdate, place from artist a where not exists (select art_id from album al where a.a_id = al.art_id);
  aname   | minit | lname |   bdate    | place
----------+-------+-------+------------+--------
 Jayanthi | G     | Rao   | 1954-06-22 | karvar
(1 row)
```

4)List of albums of Hamsalekha

```
select title, albumyear_date, genre
from album
 where art_id in (select a_id from artist where aname in('Hamsalekha'));
```

```
ghj=# select title, albumyear_date, genre from album where art_id in (select a_id from artist where aname in('Hamsalekha'));
   title    | albumyear_date |      genre
------------+----------------+-----------------
 Premachari | 1997-08-26     | flock
 Premalokha | 1988-09-26     | romantic
 Agumbe     | 1970-09-27     | romantic
(3 rows)
```

5)List the details of user who have played songs

```
select uname, email, age, p_id
from user_details
where email in ( select user_email from playlist);
```

```
ghj=# select uname, email, age, p_id from user_details where email in ( select user_email from playlist);
    uname     |         email          | age | p_id
--------------+------------------------+-----+------
 RajeshKrishnan | rajesh@gmail.com       |  55 |   11
 Vijayprakash   | vijayprakash@gmail.com |  50 |   22
 Shreyagoshal   | shreyagoshal@gmail.com |  35 |   33
 Chaithra       | chaithra@gmail.com     |  34 |   44
(4 rows)
```

Agregated function queries:

1)Count the number of alumbs for each artist

select art_id, count(*) no_of_albums from album group by art_id;

```
ghj=# select art_id, count(*) no_of_albums from album group by art_id;
 art_id | no_of_albums
--------+--------------
 666777 |            2
 111222 |            1
 444555 |            1
 777888 |            1
 222333 |            3
 333444 |            1
(6 rows)
```

2) To count number of songs in perticular album , avarage, minnimum and maximum duration played in each album.
select al_id, count(*) No_of_songs, avg(duration), min(duration), max(duration)
from song
group by al_id;

```
ghj=# select al_id, count(*) No_of_songs, avg(duration), min(duration), max(duration) from song group by al_id;
 al_id | no_of_songs |        avg         | min | max
-------+-------------+--------------------+-----+-----
     5 |           3 | 5.1666666666666667 | 4.5 | 5.8
     2 |           3 | 4.4000000000000000 | 3.6 | 5.0
     4 |           2 | 5.2000000000000000 | 4.2 | 6.2
     8 |           3 | 5.5000000000000000 | 4.6 | 6.4
(4 rows)
```

3)As nested query to list the song titles whose duration is greater than average duration.

```
select song_title,duration
from song s
 where duration > (select avg(duration)
                        from song
                        where al_id = s.al_id);
```

```
postgres=# \c ghj;
You are now connected to database "ghj" as user "postgres".
ghj=# select song_title,duration from song s where duration > (select avg(duration) from song where al_id = s.al_id);
   song_title    | duration
-----------------+----------
 Yaripremachari |      5.8
 Hoovantha      |      5.2
 Jaisriram      |      6.4
 Papamere       |      6.2
 Lifuistene     |      4.6
 Udisuve        |      5.0
(6 rows)


ghj=#
```

Outer join queries

1)List the playlist name, id, user name, for user email matching by left outer join(only matching rows)

select p.playlistname, p.play_id, p.user_email, u.uname from playlist p left outer join user_details u on p.user_email = u.email;

```
ghj=# select p.playlistname, p.play_id, p.user_email, u.uname from playlist p left outer join user_details u on p.user_email = u.email;
  playlistname  | play_id |       user_email       |     uname
----------------+---------+------------------------+---------------
 Hamsalekhaplay |      11 | rajesh@gmail.com       | RajeshKrishnan
 Nagendraplay   |      22 | vijayprakash@gmail.com | Vijayprakash
 Yuvashankan    |      33 | shreyagoshal@gmail.com | Shreyagoshal
 Yougarajplay   |      44 | chaithra@gmail.com     | Chaithra
(4 rows)
```

2)List the album name, id, artist name, for matching artist id by full outer join(lists all the rows on both the sides as wellas matching rows)

select a.aname, al.album_id, al.title
from artist a full outer join album al on a.a_id = al.art_id;

```
ghj=# select a.aname, al.album_id, al.title from artist a full outer join album al on a.a_id = al.art_id;
     aname      | album_id |    title
----------------+----------+-------------
 MichaelJackson |        1 | Thriller
 Yougaraj       |        2 | Pancharangi
 Yuvashankan    |        4 | Sathyagragh
 Hamsalekha     |        5 | Premachari
 Baskaran       |        6 | songofyouth
 Baskaran       |        7 | bodyguard
 Nagendra       |        8 | gaana
 Hamsalekha     |        9 | Premalokha
 Hamsalekha     |       10 | Agumbe
 Jayanthi       |          |
(10 rows)
```

Conclusion

The above data base management system of music player system, which enables to offer more convenience as well as efficiency to access data from database.
DBMS which consists of tools making data, data saving and data manipulating an easier task. The stored information can be used in the future. The user will find it is easy in this automated system rather than using the manual writing system. The system contains the database where all the information will be stored safely.

The above system has the capability of storing the enough details of an artist and his contribution towards the music field. It also allows to know more about the artist current position in the music field through his awards been won. Totally above system helps in maintaining the structured data about the music artists and their albums easily.

Limitations and future enhancements:
The above data base system of music player unable to store more details about the playlist like about its tag and so on. And also we can measure the song based on its rating too, which is not there in the above system.
So the future enhancement would be making able to store more details about the playlist tags and ratings of the songs so on.
And also about the security of stored data should be enhanced in the future.