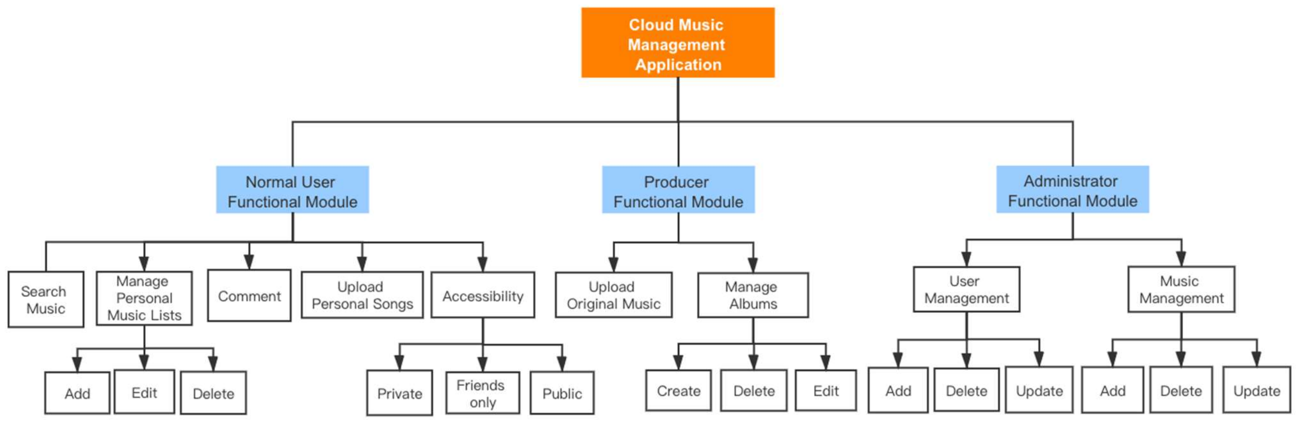
Music Management Application

I. MEMBERS

Name	NUID	Work done by Teammates
Bo Liu	001057068	Modified tables and redefined ER diagram, Designed Security and Business Rules
Tianyu Wei	001371254	Modified tables and redefined ER diagram, Designed Security and Business Rules
Pooja Vaikos	001065096	Modified tables and redefined ER diagram, Designed Security and Business Rules
Vraj Gandhi	001443543	Modified tables and redefined ER diagram, Designed Security and Business Rules

II. PROBLEM STATEMENT

Nowadays, many music applications orient to listeners only, which leads to the lack of interaction between producers and normal users. Therefore, this project is aimed to bridge the gap and enrich the functions of music application.



III. ROLE PARTITION

Normal user: refers to all listeners using this application, have the lowest accessibility.

Producer: refers to professional singers and composers. Application enables them.

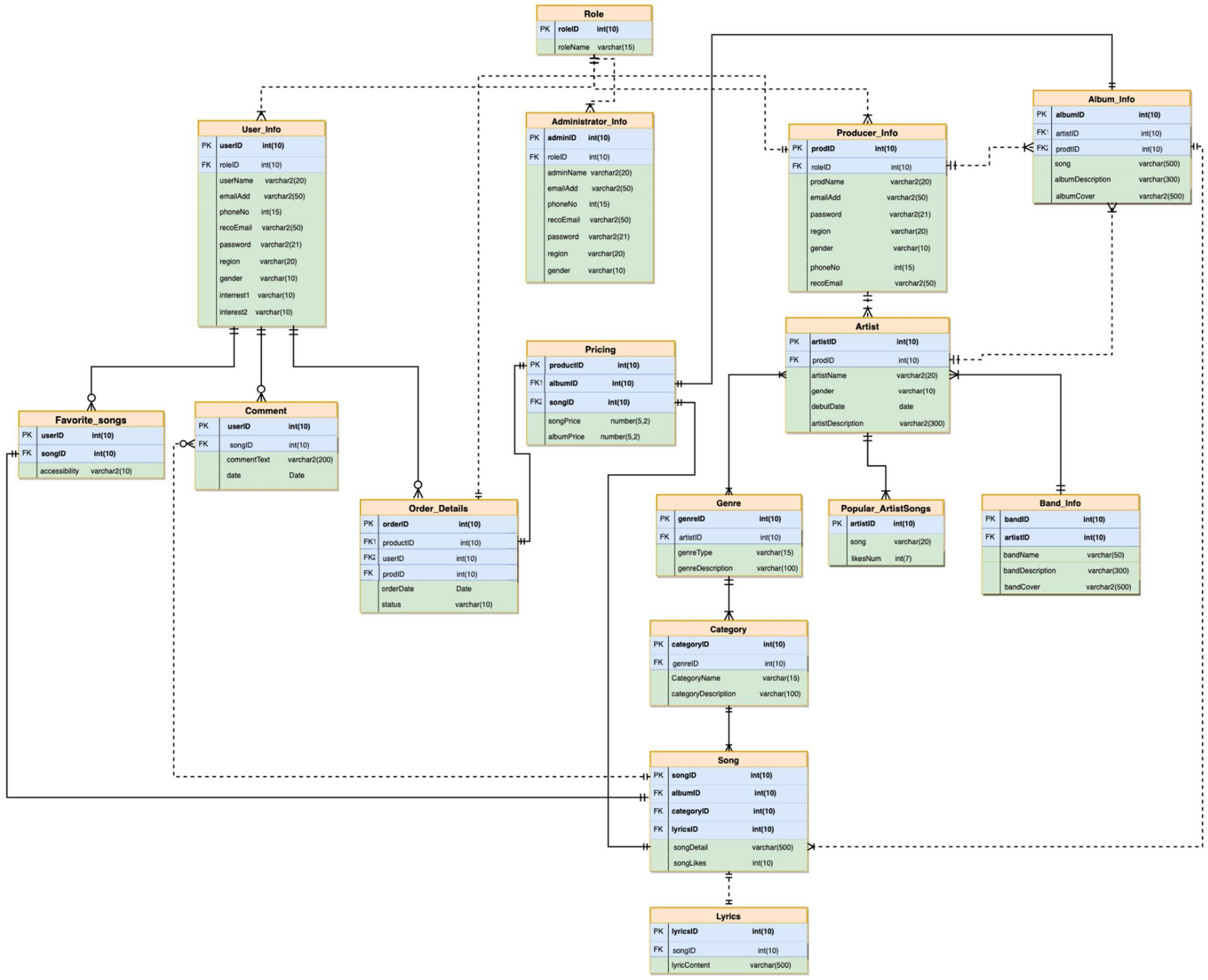
Administrator: responsible for all account management, have the highest Accessibility.

IV. OBJECTIVES

a. Terminate the conventional way of music listening and design creative functions to improve user experience. User can be a listener, commentator and producer simultaneously.
b. Professional producers can post new music and album in a more efficient and convenient way.
c. Users can share music lists they love to someone else or keep them in private.

V. E-R DIAGRAM

a. Logical Model



VI. TABLE STRUCTURES

a. Role Table

Attributes	Type	Default Value	NULL?	Description
RoleID(PK)	INT(10)	No	No	Primary key ID of every role in this application. 1 refers to “User”,2 refers to “Producer”,3 refers to “Administrator”
roleName	VARCHAR2(15)	No	No	The name of each role.

b. User_Info Table

Attributes	Type	Default Value	NULL?	Description
UserID(PK)	Int(10)	No	No	User ID of User
Roles_ID(FK)	Int(10)	1	No	Role ID of User
userName	Varchar2(20)	No	No	User Name
emailAdd	Varchar2(20)	No	No	Email Id of User
recoEmail	Varchar2(50)	No	No	User e-mail for recovery
Password	Varchar2(20)	No	No	Password
Phone No.	Int(12)	No	Yes	Phone no. of User
region	Varchar2(20)	No	Yes	Region that user is belong to
gender	Varchar2(10)	No	Yes	User’s gender
interest1	Varchar(10)	No	Yes	Interests user has
interest2	Varchar(10)	No	Yes	Interests user has

c. Favorite_Songs Table

Attributes	Type	Default Value	NULL?	Description
UserID(PK)	Int(10)	No	No	User ID
roles_ID(FK)	Int(10)	1	No	Role ID of User
accessibility	Varchar2(10)	No	No	Public, private, Friends only

d. Comment Table

Attributes	Type	Default Value	NULL?	Description
UserID(PK)	Int(10)	No	No	User ID
Song ID(FK)	Int(10)	No	No	Song ID
CommentsText	Varchar2(200)	No	No	Comments Text
Date	Date	No	No	Date of the posted Comment

e. Producer_Info Table

Attributes	Type	Default Value	NULL?	Description
ProdID(PK)	INT(10)	No	No	Primary key The unique ID of each producer.
RoleID(FK)	VARCHAR2(15)	2	No	Foreign key The role ID of producer company.
prodName	VARCHAR2(20)	No	No	Brand name of each producer.
prodEmail	VARCHAR2(50)	No	No	Email information of each producer.
Recovery Email	Varchar2(20)	No	No	Recover email in case of forgetting the password
Password	Varchar2(20)	No	No	Password
region	Varchar2(20)	No	Yes	Region that producer is belong to
gender	Varchar2(10)	No	Yes	Producer's gender

f. Artist Table

Attributes	Type	Default Value	NULL?	Description
ArtistID(PK)	INT(10)	No	No	Primary key The unique ID of each artist.
BandID(FK)	INT(10)	No	Yes	Foreign key The band ID of each artist(if not in a band, the band ID is null)
ProdID(FK)	VARCHAR2(20)	No	No	Foreign key

				The producer ID of each artist.
artistName	VARCHAR2(20)	No	No	The personal name of each artist.
gender	VARCHAR(10)	No	Yes	The gender of the artist 0 refers to “man” 1 refers to “female” 2 refers to “others”
debutDate	DATE	No	No	The time of artist having first performance to public.
artistDescription	VARCHAR2(300)	No	Yes	Introduction of each artist.

g. Genre Table

Attributes	Type	Default Value	NULL?	Description
GenreID(PK)	INT(10)	No	No	Primary key The unique ID for each song genre. 0 refers to “Indian” 1 refers to “American” 2 refers to “Japanese” 3 refers to “Chinese” Etc.
ArtistID(FK)	INT(10)	No	No	Foreign key The artist ID of each artist.
genreType	VARCHAR(15)	No	No	The genre name(Indian, Chinese,etc)
genreDescription	VARCHAR2(100)	No	Yes	The description of each genre type.

h.Category Table

Attributes	Type	Default Value	NULL?	Description
categoryID(PK)	INT(10)	No	No	Tells us about the mood of song
genreID(FK)	INT(10)	No	No	Links to genre it comes into
CategoryName	VARCHAR(15)	No	No	Eg: Romantic, Sad, Motivation, Happy etc

CategoryDescription	VARCHAR (100)	No	Yes	Why the song is into that category

i. Song Table

Attributes	Type	Default Value	NULL?	Description
SongID(PK)	INT(10)	No	No	ID of songs
AlbumID(FK)	INT(10)	No	No	AlbumID of the song belongs to
CategoryID(FK)	INT(10)	No	No	Which category song belongs to
LyricsID(FK)	INT(10)	No	No	LyricsID of the song belongs to
SongDetail	VARCHAR(20)	No	No	Details information of song
songLikes	INT(10)	No	Yes	The amount of likes the song gets

j. Album_Info Table

Attributes	Type	Default Value	NULL?	Description
AlbumID(PK)	INT(10)	No	No	Unique value that links to artist and producer
ArtistID(FK1)	INT(10)	No	No	Unique ID to show the artist in that album
Producer(FK2)	INT(10)	No	No	Unique ID to show the producer of that album
Song	VARCHAR(20)	No	No	Name of the song
albumDescription	VARCHAR(300)	No	No	Brief information of making of the album
albumCover	VARCHAR2(500)	No	Yes	URL link for video

k.Band_Info Table

Attributes	Type	Default Value	NULL?	Description
BandID(PK)	INT(10)	No	No	Unique ID for different bands
BandName	Varchar(50)	No	No	Name of the band
bandDescription	VARCHAR(300)	No	Yes	How was it formed
BandCover	VARCHAR(500)	No	Yes	URL link for Vedio

l. Administrator_Info Table

Attributes	Type	Default Value	NULL?	Description
AdminID(PK)	Int(10)	No	No	The ID of administrator in active
RoleID(FK)	Int(10)	3	No	Represents the role of user

userName	Varchar2(20)	No	No	The name of administrator
emailAdd	Varchar2(50)	No	No	E-mail of administrator
phoneNo	Int(15)	No	Yes	Phone number of administrator
recoEmail	Varchar2(50)	No	No	The E-mail address for recovery use
password	Varchar2(50)	No	No	The password for administrator to access the system
region	Varchar(20)	No	No	Country the administrator is originally from
gender	Varchar(10)	No	Yes	The gender of administrator

m. Order Details Table:

Attributes	Type	Default Value	NULL?	Description
OrderID(PK)	Int(10)	No	No	The order Id of purchased items
ProductID(FK)	Int(10)	No	No	Product ID of charged items
UserID(FK)	Int(10)	No	No	User id of the user that has purchased an item
ProdID(FK)	Int(10)	No	No	Producer ID who has purchased/sold any item
orderDate	date	No	No	Order date
Status	Varchar(10)	No	No	Processing, pending, completed

n. Popular_ArtistSongs Table

Attributes	Type	Default Value	NULL?	Description
ArtistID(PK)	Int(10)	No	No	The ID of artist who is in active
song	Varchar(20)	No	No	The name of song that is listing for
likesNum	Int(7)	No	No	The number of likes the song gets

o.Pricing Table

Attributes	Type	Default Value	NULL?	Description
ProductID(PK)	Int(10)	No	No	The ID of new transaction
AlbumID(FK)	Int(10)	No	No	The ID of album

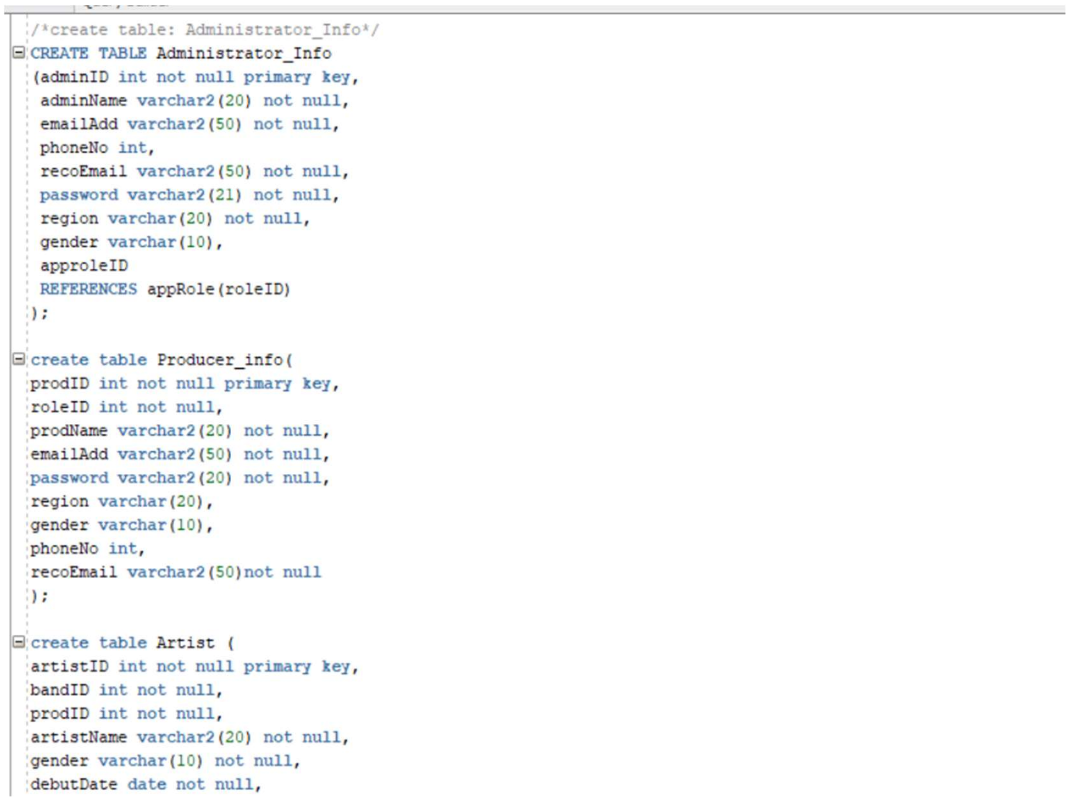
SongID(FK)	Int(10)	No	No	Song ID for any Song
songPrice	Number(5,2)	0	No	Price that can be charged for a song
albumPrice	Number(5,2)	0	No	Price that can be charged for an album

p. Lyrics Table:

Attributes	Type	Default Value	NULL?	Description
LyricsID(PK)	Int(10)	No	No	The ID for new lyrics
SongID(FK)	Int(10)	No	No	Song ID for any Song
lyricsContent	Varchar(500)	No	No	Lyrics of any song

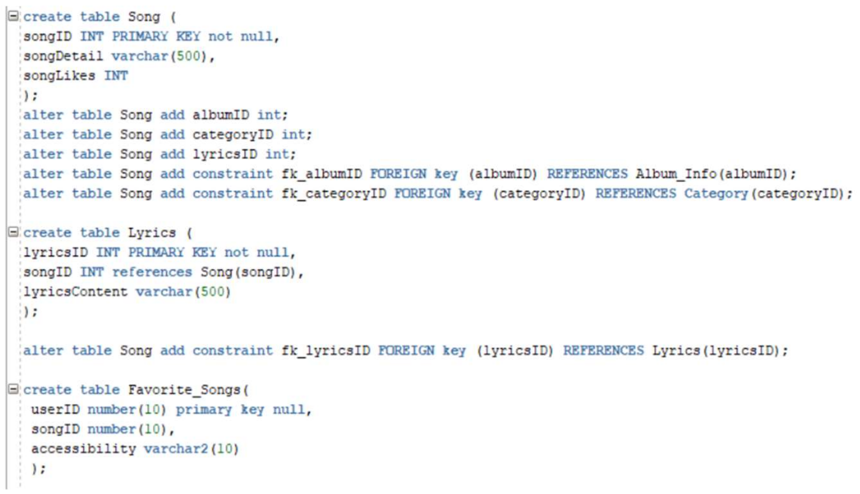
VII) Create tables and contraints

We create tables using ‘create’ syntax to build tables. The constraints include foreign keys and primary keys which were created along with tables.



```
/*create table: Administrator_Info*/
CREATE TABLE Administrator_Info
(adminID int not null primary key,
 adminName varchar2(20) not null,
 emailAdd varchar2(50) not null,
 phoneNo int,
 recoEmail varchar2(50) not null,
 password varchar2(21) not null,
 region varchar(20) not null,
 gender varchar(10),
 approleID
 REFERENCES appRole(roleID)
);

create table Producer_info(
prodID int not null primary key,
roleID int not null,
prodName varchar2(20) not null,
emailAdd varchar2(50) not null,
password varchar2(20) not null,
region varchar(20),
gender varchar(10),
phoneNo int,
recoEmail varchar2(50)not null
);

create table Artist (
artistID int not null primary key,
bandID int not null,
prodID int not null,
artistName varchar2(20) not null,
gender varchar(10) not null,
debutDate date not null,
```

‘References’ as above represents a foreign key links to roleID in appRole table. ‘Not null’ represents the attribute it’s under could not be set as null. ‘Primary key’ represents that the attribute it’s under is a primary key which could be referred to a foreign key from another table.

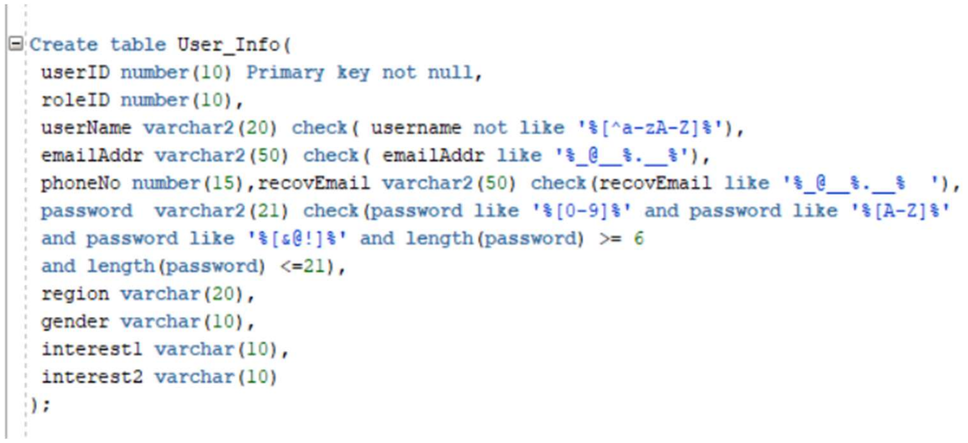


```
create table Song (
songID INT PRIMARY KEY not null,
songDetail varchar(500),
songLikes INT
);
alter table Song add albumID int;
alter table Song add categoryID int;
alter table Song add lyricsID int;
alter table Song add constraint fk_albumID FOREIGN key (albumID) REFERENCES Album_Info(albumID);
alter table Song add constraint fk_categoryID FOREIGN key (categoryID) REFERENCES Category(categoryID);

create table Lyrics (
lyricsID INT PRIMARY KEY not null,
songID INT references Song(songID),
lyricsContent varchar(500)
);

alter table Song add constraint fk_lyricsID FOREIGN key (lyricsID) REFERENCES Lyrics(lyricsID);

create table Favorite_Songs(
 userID number(10) primary key null,
 songID number(10),
 accessibility varchar2(10)
 );
```

As the script shows above, we use 'alter' syntax to create new foreign keys in child tables to link two associated tables after those tables have already been created.

VIII) Validations

It is essential to add validation conditions in the database we created to help users validate input data into the right format. Block out the one which is not fit for the attribute or has negative effect for the system performance.
In this project, we use several ways to make this goal well-functioned.

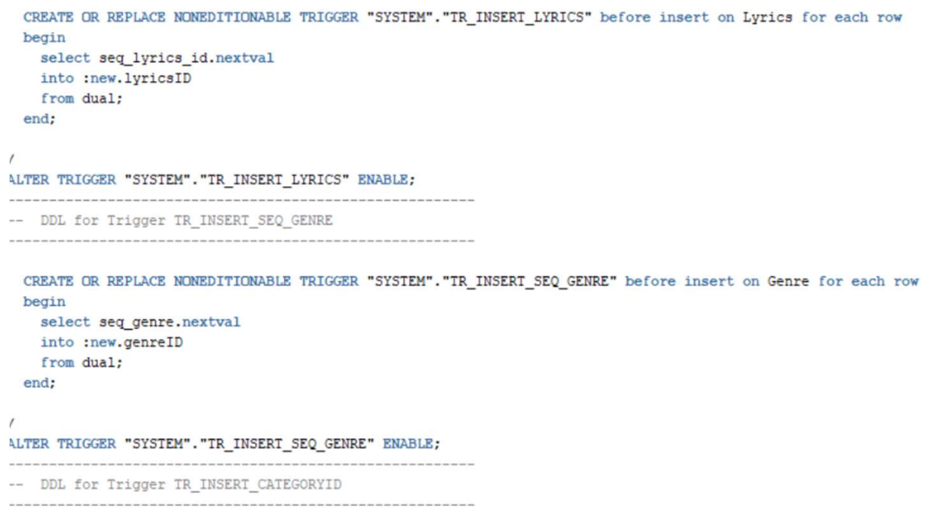


```
Create table User_Info(
 userID number(10) Primary key not null,
 roleID number(10),
 userName varchar2(20) check( username not like '%[^a-zA-Z]%'),
 emailAddr varchar2(50) check( emailAddr like '%_@__%.__%'),
 phoneNo number(15),recovEmail varchar2(50) check(recovEmail like '%_@__%.__%  '),
 password  varchar2(21) check(password like '%[0-9]%' and password like '%[A-Z]%'
 and password like '%[&@!]%' and length(password) >= 6
 and length(password) <=21),
 region varchar(20),
 gender varchar(10),
 interest1 varchar(10),
 interest2 varchar(10)
);
```

The statement above shows one of the several ways to valid user input. Use 'not like' syntax to fit the pattern that we set using different characters.
Attribute 'emailAddr' is validated using regular expression to match email pattern.

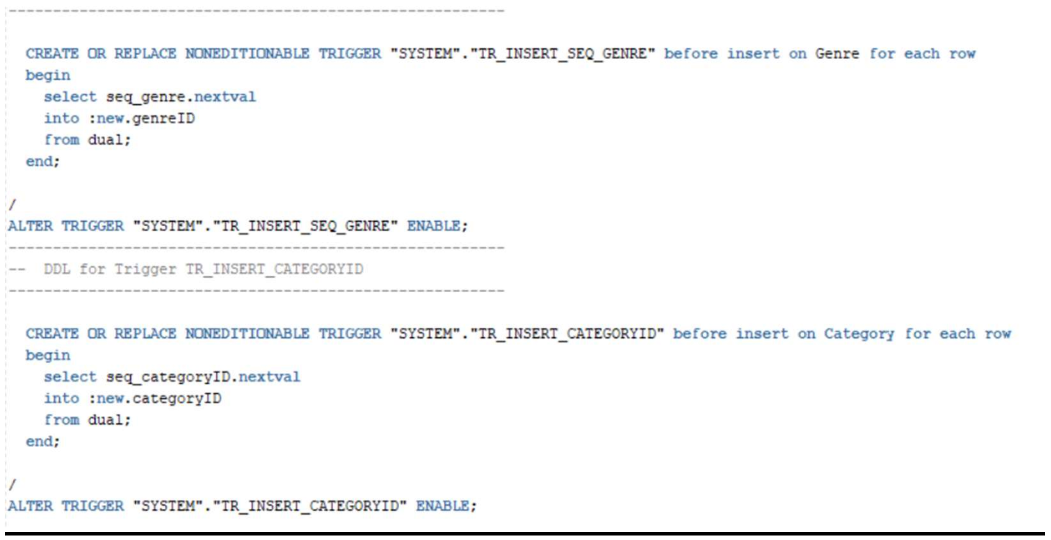
IX) Trigger

In order to load data more efficiently and accurately, we add triggers in several tables to create ID automatically which is the crucial part to generate ID for each piece of inserted data with no conflict happens along with it.
To implement this idea, we create triggers as below.



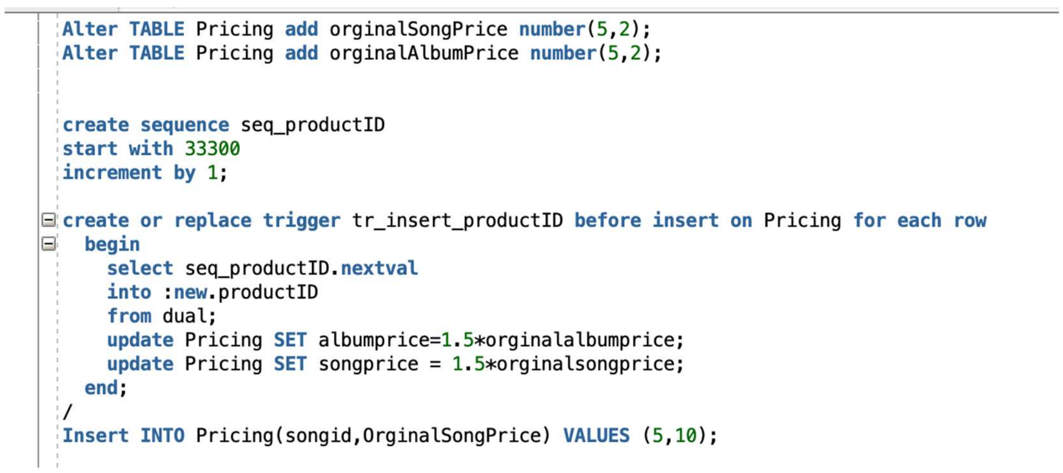
```
  CREATE OR REPLACE NONEDITIONABLE TRIGGER "SYSTEM"."TR_INSERT_LYRICS" before insert on Lyrics for each row
  begin
    select seq_lyrics_id.nextval
    into :new.lyricsID
    from dual;
  end;

/
ALTER TRIGGER "SYSTEM"."TR_INSERT_LYRICS" ENABLE;
--------------------------------------------------------
--  DDL for Trigger TR_INSERT_SEQ_GENRE
--------------------------------------------------------

  CREATE OR REPLACE NONEDITIONABLE TRIGGER "SYSTEM"."TR_INSERT_SEQ_GENRE" before insert on Genre for each row
  begin
    select seq_genre.nextval
    into :new.genreID
    from dual;
  end;

/
ALTER TRIGGER "SYSTEM"."TR_INSERT_SEQ_GENRE" ENABLE;
--------------------------------------------------------
--  DDL for Trigger TR_INSERT_CATEGORYID
--------------------------------------------------------
```

The trigger shows above fires when producer trying to add a new lyric and genre record and generate a brand new ID for each record in descending order.



```
--------------------------------------------------------

  CREATE OR REPLACE NONEDITIONABLE TRIGGER "SYSTEM"."TR_INSERT_SEQ_GENRE" before insert on Genre for each row
  begin
    select seq_genre.nextval
    into :new.genreID
    from dual;
  end;

/
ALTER TRIGGER "SYSTEM"."TR_INSERT_SEQ_GENRE" ENABLE;
--------------------------------------------------------
--  DDL for Trigger TR_INSERT_CATEGORYID
--------------------------------------------------------

  CREATE OR REPLACE NONEDITIONABLE TRIGGER "SYSTEM"."TR_INSERT_CATEGORYID" before insert on Category for each row
  begin
    select seq_categoryID.nextval
    into :new.categoryID
    from dual;
  end;

/
ALTER TRIGGER "SYSTEM"."TR_INSERT_CATEGORYID" ENABLE;
```

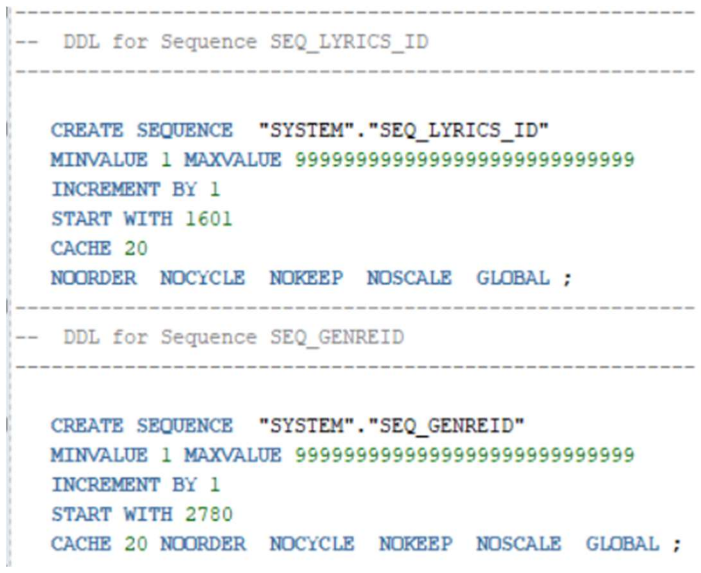
The trigger above will execute as the genre and category table inserting request received which automatically generate a new ID for the newly came record.

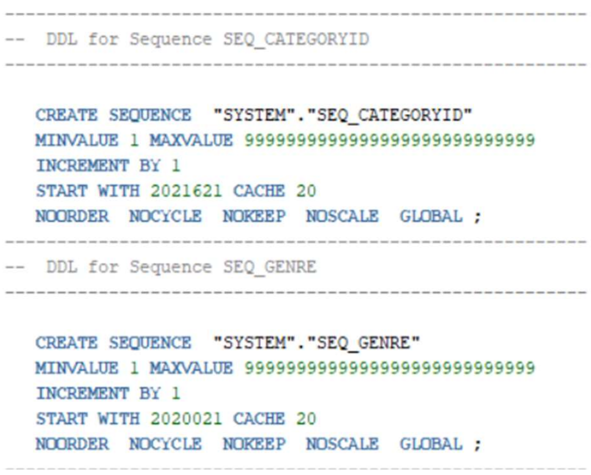

```
Alter TABLE Pricing add orginalSongPrice number(5,2);
Alter TABLE Pricing add orginalAlbumPrice number(5,2);


create sequence seq_productID
start with 33300
increment by 1;

create or replace trigger tr_insert_productID before insert on Pricing for each row
  begin
    select seq_productID.nextval
    into :new.productID
    from dual;
    update Pricing SET albumprice=1.5*orginalalbumprice;
    update Pricing SET songprice = 1.5*orginalsongprice;
  end;
/
Insert INTO Pricing(songid,OrginalSongPrice) VALUES (5,10);
```

The trigger above automatically generate the price which would be 50% up from its original price to regular price for customer as the original price is the sell price for producer.

X) Sequence

In terms of creating an order-based ID for each table, we set triggers to make it happen. However, we still need a way to regulate the number it changes for each roll. Sequence bridges this gap by adding a constraint for the trigger that we added to keep it running in the regulation we set.


```
--------------------------------------------------------
--  DDL for Sequence SEQ_LYRICS_ID
--------------------------------------------------------

   CREATE SEQUENCE  "SYSTEM"."SEQ_LYRICS_ID"
   MINVALUE 1 MAXVALUE 9999999999999999999999999999
   INCREMENT BY 1
   START WITH 1601
   CACHE 20
   NOORDER  NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
--------------------------------------------------------
--  DDL for Sequence SEQ_GENREID
--------------------------------------------------------

   CREATE SEQUENCE  "SYSTEM"."SEQ_GENREID"
   MINVALUE 1 MAXVALUE 9999999999999999999999999999
   INCREMENT BY 1
   START WITH 2780
   CACHE 20 NOORDER  NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
```

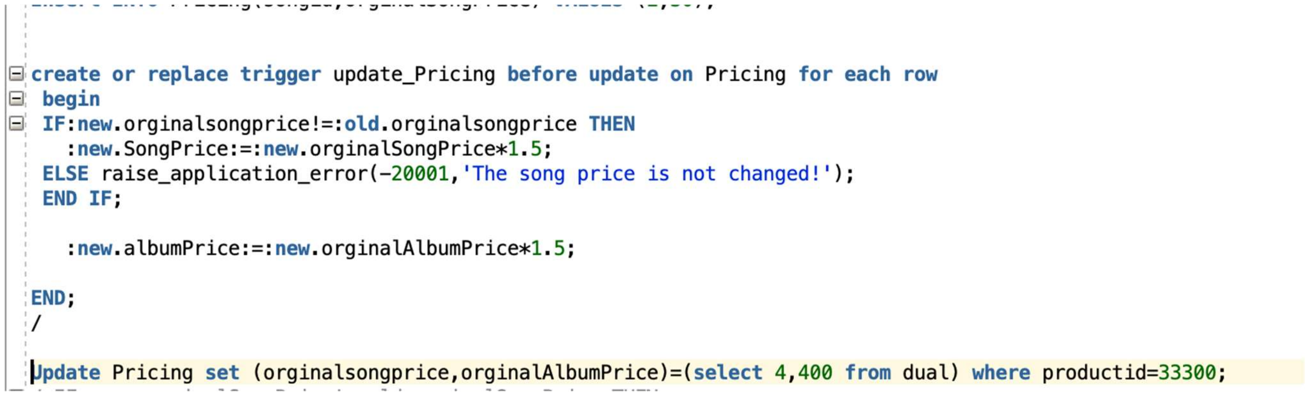


```
--------------------------------------------------------
--  DDL for Sequence SEQ_CATEGORYID
--------------------------------------------------------

   CREATE SEQUENCE  "SYSTEM"."SEQ_CATEGORYID"
   MINVALUE 1 MAXVALUE 9999999999999999999999999999
   INCREMENT BY 1
   START WITH 2021621 CACHE 20
   NOORDER  NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
--------------------------------------------------------
--  DDL for Sequence SEQ_GENRE
--------------------------------------------------------

   CREATE SEQUENCE  "SYSTEM"."SEQ_GENRE"
   MINVALUE 1 MAXVALUE 9999999999999999999999999999
   INCREMENT BY 1
   START WITH 2020021 CACHE 20
   NOORDER  NOCYCLE  NOKEEP  NOSCALE  GLOBAL ;
```

It shows above that we add sequence constraint in global scope. The spot that sequence runs would generate a specific pattern for the host that execute to manage the number. We create sequences seq_lyrics_id to apply a custom range and regulation for other controller to automatically generate id.
Sequence Seq_genreid, Seq_categoryid, Seq_genre serves the similar purpose to give controller a custom rule to follow up with.

XI) Exception tracking

To better tracking and debugging errors in our database, we define an exception throwing method for developer to know which spot has issue in runtime.



```
create or replace trigger update_Pricing before update on Pricing for each row
 begin
 IF:new.orginalsongprice!=:old.orginalsongprice THEN
   :new.SongPrice:=:new.orginalSongPrice*1.5;
 ELSE raise_application_error(-20001,'The song price is not changed!');
 END IF;

   :new.albumPrice:=:new.orginalAlbumPrice*1.5;

END;
/

Update Pricing set (orginalsongprice,orginalAlbumPrice)=(select 4,400 from dual) where productid=33300;
```

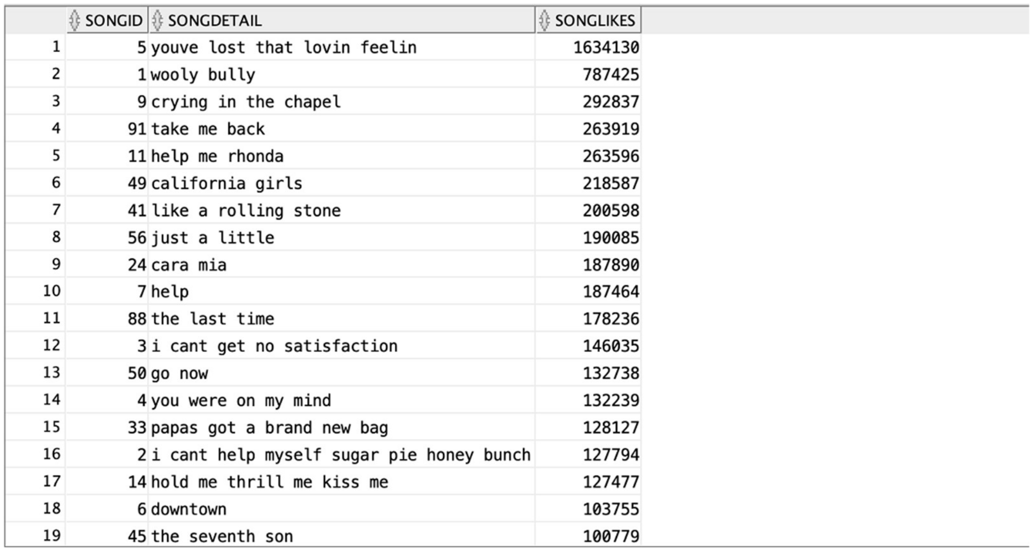
As scripts above, it will throw an exception when it doesn't satisfy the conditions which will show in the console window. It provides an easy way to handle the issues when database running in the real-world.

XII) Scripts execution

1. List the popular songID from top to low:

Select songID, songLikes
from Song
order by songLikes
desc;

Result:

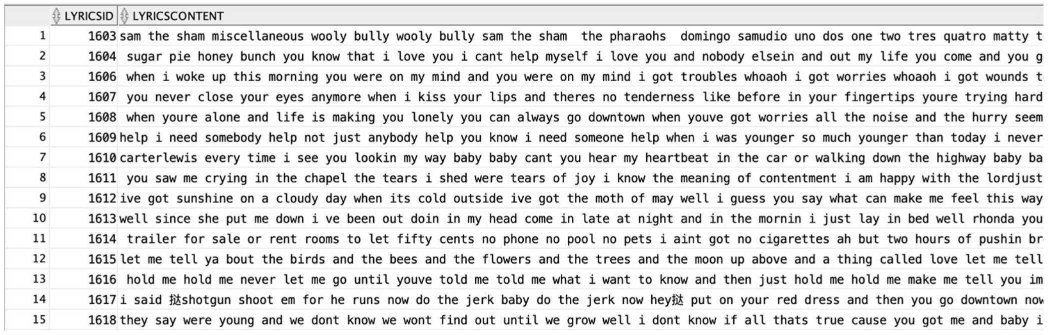


	SONGID	SONGDETAIL	SONGLIKES
1	5	youve lost that lovin feelin	1634130
2	1	wooly bully	787425
3	9	crying in the chapel	292837
4	91	take me back	263919
5	11	help me rhonda	263596
6	49	california girls	218587
7	41	like a rolling stone	200598
8	56	just a little	190085
9	24	cara mia	187890
10	7	help	187464
11	88	the last time	178236
12	3	i cant get no satisfaction	146035
13	50	go now	132738
14	4	you were on my mind	132239
15	33	papas got a brand new bag	128127
16	2	i cant help myself sugar pie honey bunch	127794
17	14	hold me thrill me kiss me	127477
18	6	downtown	103755
19	45	the seventh son	100779

2. Select the lyrics details where lyrics characters are more than 300 words:

Select LyricsID, lyricscontent
from lyrics
where length(lyricsContent) > 300;

Result:



	LYRICSID	LYRICSCONTENT
1	1603	sam the sham miscellaneous wooly bully wooly bully sam the sham the pharaohs domingo samudio uno dos one two tres quatro matty t
2	1604	sugar pie honey bunch you know that i love you i cant help myself i love you and nobody elsein and out my life you come and you g
3	1606	when i woke up this morning you were on my mind and you were on my mind i got troubles whoaoh i got worries whoaoh i got wounds t
4	1607	you never close your eyes anymore when i kiss your lips and theres no tenderness like before in your fingertips youre trying hard
5	1608	when youre alone and life is making you lonely you can always go downtown when youve got worries all the noise and the hurry seem
6	1609	help i need somebody help not just anybody help you know i need someone help when i was younger so much younger than today i never
7	1610	carterlewis every time i see you lookin my way baby baby cant you hear my heartbeat in the car or walking down the highway baby ba
8	1611	you saw me crying in the chapel the tears i shed were tears of joy i know the meaning of contentment i am happy with the lordjust
9	1612	ive got sunshine on a cloudy day when its cold outside ive got the moth of may well i guess you say what can make me feel this way
10	1613	well since she put me down i ve been out doin in my head come in late at night and in the mornin i just lay in bed well rhonda you
11	1614	trailer for sale or rent rooms to let fifty cents no phone no pool no pets i aint got no cigarettes ah but two hours of pushin br
12	1615	let me tell ya bout the birds and the bees and the flowers and the trees and the moon up above and a thing called love let me tell
13	1616	hold me hold me never let me go until youve told me told me what i want to know and then just hold me hold me make me tell you im
14	1617	i said 挞shotgun shoot em for he runs now do the jerk baby do the jerk now hey挞 put on your red dress and then you go downtown now
15	1618	they say were young and we dont know we wont find out until we grow well i dont know if all thats true cause you got me and baby i

3. Dates Functions:

select orderDate, add_months(orderDate, 4)
from order_details;

4.List user details who have bought more than one sone:

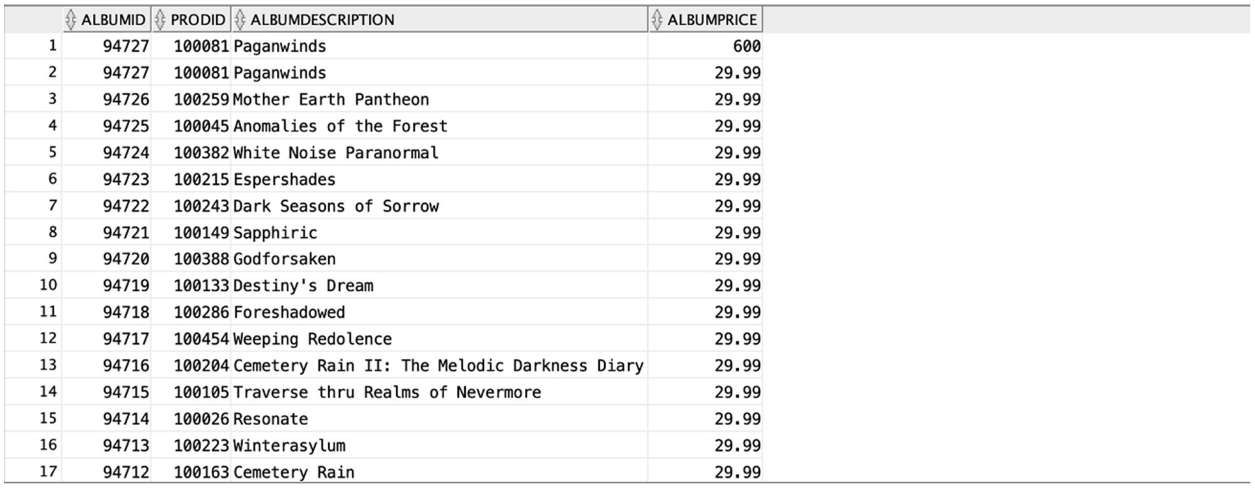
select userID, userName

```
from user_Info u
where 1< (select orderID, OrderDate, Status from Order_Details where userID = u.userID);
```

5. See the album AlbumDescription and its price;

```
Select album_info.albumID, album_info.PRODID, album_info.AlbumDescription, pricing, albumprice
from album_info
full outer join Pricing.albumID = album_info.albumID;
```

Result:

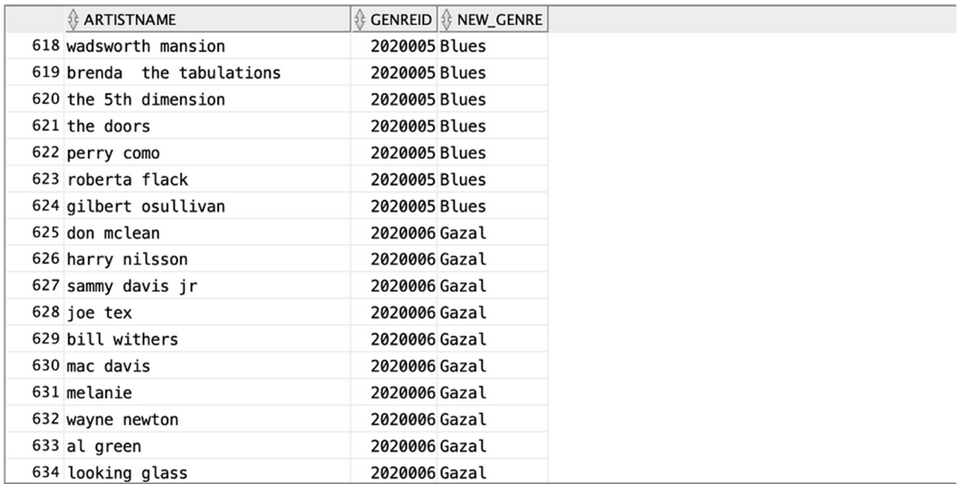


	ALBUMID	PRODID	ALBUMDESCRIPTION	ALBUMPRICE
1	94727	100081	Paganwinds	600
2	94727	100081	Paganwinds	29.99
3	94726	100259	Mother Earth Pantheon	29.99
4	94725	100045	Anomalies of the Forest	29.99
5	94724	100382	White Noise Paranormal	29.99
6	94723	100215	Espershades	29.99
7	94722	100243	Dark Seasons of Sorrow	29.99
8	94721	100149	Sapphiric	29.99
9	94720	100388	Godforsaken	29.99
10	94719	100133	Destiny's Dream	29.99
11	94718	100286	Foreshadowed	29.99
12	94717	100454	Weeping Redolence	29.99
13	94716	100204	Cemetery Rain II: The Melodic Darkness Diary	29.99
14	94715	100105	Traverse thru Realms of Nevermore	29.99
15	94714	100026	Resonate	29.99
16	94713	100223	Winterasylum	29.99
17	94712	100163	Cemetery Rain	29.99

6. List Genre Type, Artist Name and decode genre 'POP' to 'Gazal'

```
select A.artistName, G.henreID
decode(G.genreType, ‘Pop’, ‘Gazal’, G.genreType) New_Genre
from Artist A
inner join Genre G
on A.genreID = G.genreID;
```

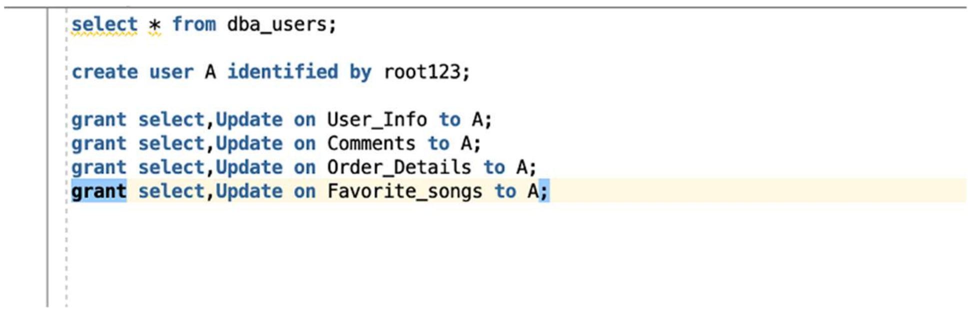
Result:



	ARTISTNAME	GENREID	NEW_GENRE
618	wadsworth mansion	2020005	Blues
619	brenda the tabulations	2020005	Blues
620	the 5th dimension	2020005	Blues
621	the doors	2020005	Blues
622	perry como	2020005	Blues
623	roberta flack	2020005	Blues
624	gilbert osullivan	2020005	Blues
625	don mclean	2020006	Gazal
626	harry nilsson	2020006	Gazal
627	sammy davis jr	2020006	Gazal
628	joe tex	2020006	Gazal
629	bill withers	2020006	Gazal
630	mac davis	2020006	Gazal
631	melanie	2020006	Gazal
632	wayne newton	2020006	Gazal
633	al green	2020006	Gazal
634	looking glass	2020006	Gazal

XII) Security control:

In case malicious user access the database and do modification or similar actions. We add several security constraints to the database to regular user privilege.



```
select * from dba_users;

create user A identified by root123;

grant select,Update on User_Info to A;
grant select,Update on Comments to A;
grant select,Update on Order_Details to A;
grant select,Update on Favorite_songs to A;
```

The above scripts shows that different user has different accessibility to the database for data security.

XII) Business Rules:

➢ **Rules for all Tables:**

- All the primary keys in the table must be of 10 digits as all the primary keys are integers - **Check Constraint.**
- Names should only contain alphabets. Numbers or special characters are not allowed- **Check Constraint.**
- All the Passwords should be a combination of one capital word, at least one number and special characters from &, @, ! and should be at least 6 characters long and no longer than 21 characters - **Check Constraint.**
- Recovery email is mandatory, and email should be in the format %@%. ___ - **Check Constraint.**
- Phone number is optional and can be NULL.
- **Pricing Table:**
    - ♦ **Attributes: songPrice and albumPrice - NOT NULL Constraint**
        - ▪ Any song is charged for $X and the entire album is charged $Y depending on the producer or admin.
        - ▪ These fields can be set to 0 as default but not negative value or NULL.

- **Artist Table:**
    - **Attribute : debutDate -CHECK Constraint**
        - It should be a past date or present date but not future one.

XIII) Security Rules:

- Permissions for each role:
    - Administrator :
        - Has a Read, Write and modify access to every table besides User_Info and Producer_Info.
        - Highest session level
    - Producer :
        - Has a read and write access to Artist, Genre, Band_Info, Album_Info and Song table. (Not Modify because he can only upload his work)
        - Second session level
    - User :
        - Has a Read and Write access to User_Info, Favorite_songs, Comments and Order_details table, has a read only access to Song table.
        - Lowest session level

XIV) Conclusion

Because of the frequent user interaction happens in the Cloud music platform which generates a huge amount of input and output demand. In order to support the gigantic demands of data flowing. We designed the whole thing database to fit the need to be more stable and efficient user experience.
To further increase user experience and being avoid lagging when facing a surge in demand, we use several technologies includes adding constraints, triggers, sequences,

regular expression. Adding validation for several attributes make the database has the capability to detect invalid input that user send to the database.
The database in this project we built can quickly map the user request to the right table it belongs to. We aim to make the flow it goes all the way through more smooth, less obstacle with shortest route.