CSU34041 INFORMATION MANAGEMENT II SQL DATABASE PROJECT

Topic Chosen: Song Management Application

CONTENTS

SECTION A

1. Application Description:

The Database is designed for a songs application. The current database application can give information related to songs ranging from the singer of a particular song to the instruments used while composing the song. There are 9 entities and each entity contains a primary key. To the fact, a primary key is a unique identifier for each record in the entity and therefore cannot be null. An attribute is noted as a foreign key when either a primary key or a unique attribute from a table is referred to another entity. The entities are made by the following assumptions for the application.
A song is sung by many singers and have many genres. Each singer sings for many albums. An album is recorded by a record label. A song uses many instruments. These form up the 9 entities where the number includes the 3 relationship entities in the database.

Song: The Song entity contains six key attributes which are all single valued. Song_Id, an integer, is the primary key while album_name, a character, is the foreign key from the entity album. Song_name is a unique attribute and hence can't be null. The other

attributes of this entity are Date_released, song Language, and Duration. The latter mentioned attributes can have duplicate values. Date_released attribute is of "DATE" Datatype and is of the format 'YYYY-MM-DD' whereas duration of the song is of "TIME(fsp)" Datatype and the format is 'hh:mm:ss.00000'.

Singer: The singer entity contains seven attributes which are single valued. The primary key is the ID of the singer which is an integer and there is no foreign key for this entity. Singer_name is considered unique. The other attributes include Country of birth of the singer, gender, date of birth, number of years active till present, and age can all have duplicate values. Years_active_till_ present attribute can have null values. Similar to the date_released attribute from the Song entity, date of birth of the singer is also a "DATE" Datatype. Finally, the age of the singer is derived from the singer's date of birth and can be called as a derived attribute.

Singer and Song are related to a relationship entity named **perform** which has two foreign keys singer_id and song_name from singer and song table respectively.

Album: Again, all are single valued attributes. Album_id, an integer, is the primary key album_name is a unique attribute. The foreign keys are record_label_id and singer_name. The other attributes number of songs in an album, length of the album, producer, and date released do not have to be unique. Similar to the previous entities, date_released attribute is of "DATE" datatype.

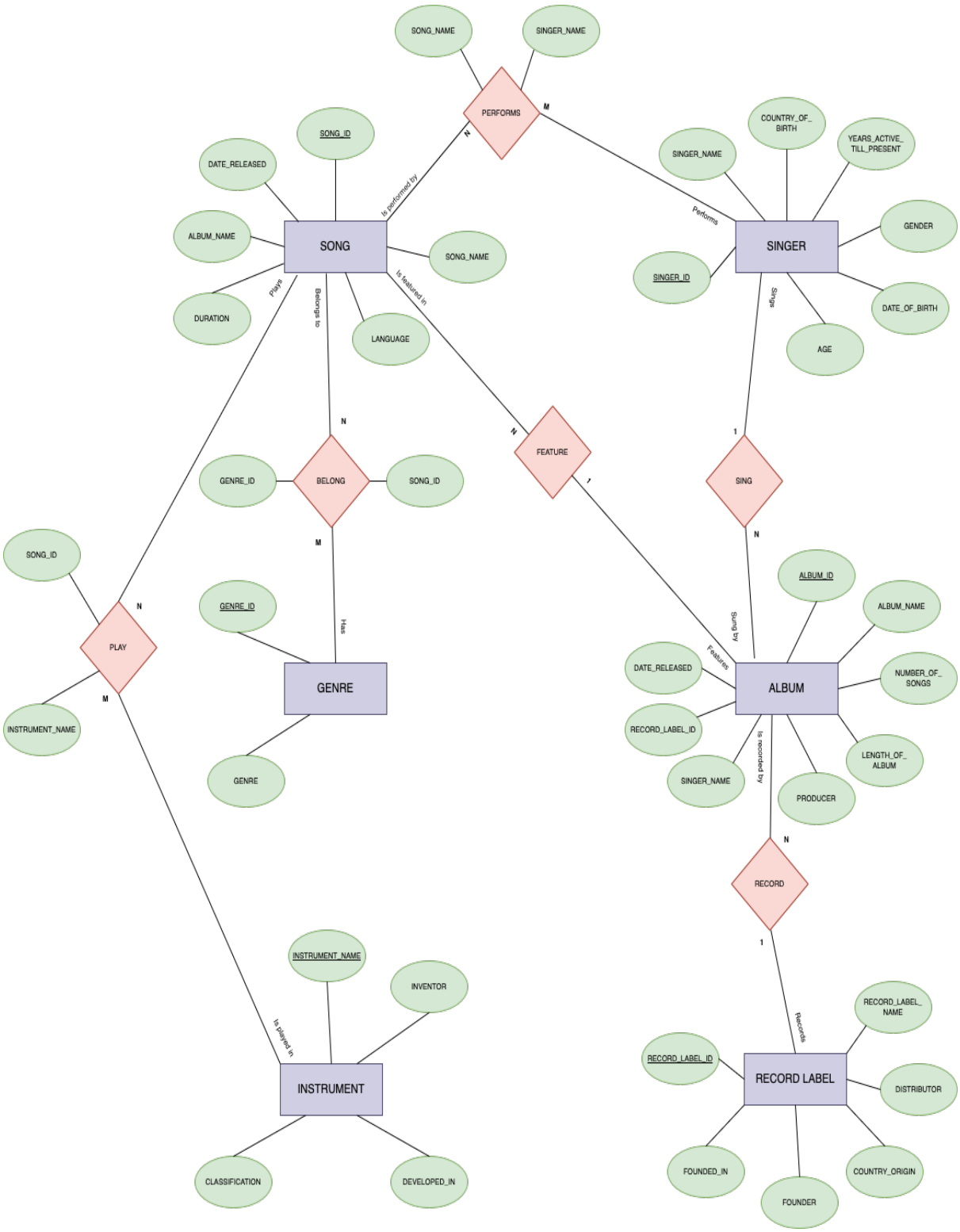
Genre: Genre is a two-attribute entity which displays the genre ID and genre name for uniqueness of genre across the database. The genre_id is the primary key and there is no foreign key for this entity and genre is unique.
Similar to singer and song, genre and song are as well related to each other by a relationship entity with two foreign attributes called "genre_id" and "song_id".

Instrument: The name of the instrument is considered to be the unique identifier for this entity and no foreign key. The classification and the inventor of the instrument which can be repetitive are of a character datatype while the year in which the instrument is developed in (developed_in) is of a "YEAR" Datatype and has a format 'YYYY'.

The relation between song and instrument is identified by a relationship entity named **play** with two foreign key attributes instrument_name and song_id from instrument and song entities respectively.

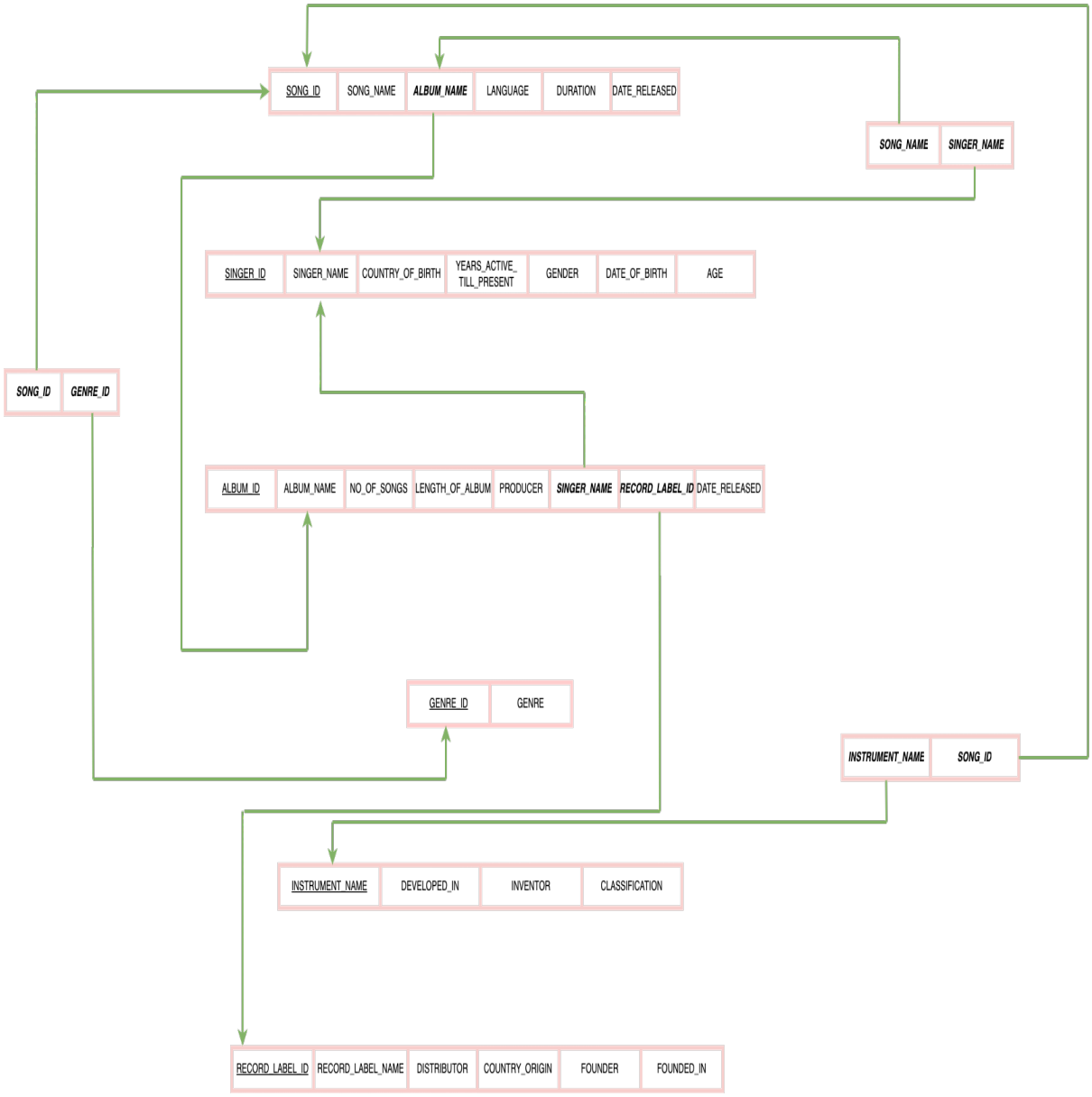
Record Label: This entity provides information on the recordings of songs. All are single valued attributes. record_label_id, is a primary key (character) and record_label_name is meant to be unique. All the other attributes have multi valued attributes.

2. Entity Relationship Diagram



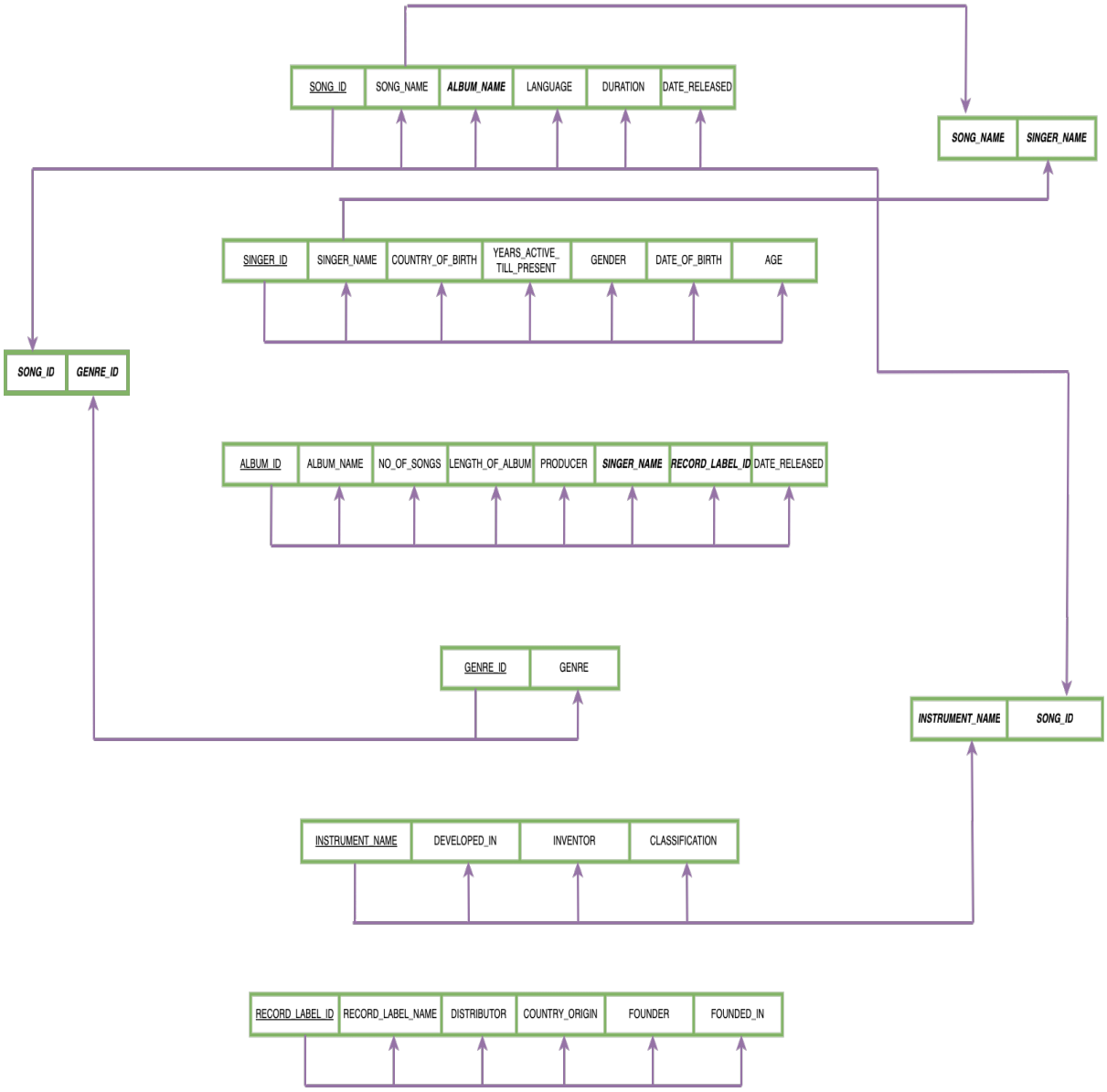
- The underlined ones are the Primary Keys
- 1:N – One to Many Relationship
- M:N – Many to Many Relationship

3. Mapping to Relational Schema



- The underlined ones are the Primary Keys
- The Bold and Italic ones are the foreign Keys

4. Functional Dependency Diagram



- The underlined ones are the Primary Keys
- The Bold and Italic ones are the foreign Keys

5. Miscellaneous/Constraints:

I have taken assumptions or might be constraints for the database application either with relation to inserting data into the tables or having a relationship between tables. This would provide an in-depth explanation of the assumptions taken.

- All the primary keys are an assumption and cannot be null
- No two songs in the song table can have the same name where song_name will be a unique key and cannot be null.
- A song can use many instruments. Each instrument can be played in many songs
- No two singers from the singer entity can have the same name and this will hence be a unique attribute and cannot be null.
- No two album names can be the same. Both album_id and album_name cannot be null values.
- A producer can produce more than one album, but an album can be produced by only one producer. If more than one producer is found for an album while inserting the data, the first one from the web is taken.
- An album can be recorded by only one record label. Each record label can record many songs.
- An album can be sung by only one singer. A singer can sing many albums.
- No two genres can have the same name and cannot be null leaving the genre to be unique and cannot.
- An inventor from instrument table can invent many instruments. Every instrument must be invented by an inventor.
- An inventor can have only one classification. A classification can be for many instruments. If many found, only the first one from the web is taken for insertion into tables.
- The instrument developed years (developed_in) from the instrument table are approx. years from that century.
- No two record label names from the record table can be matched and hence they are unique and cannot be null.
- A founder can only have many record label. Every record label must have a founder.
- A distributor can be a distributor for many record label names. Every record label must have a distributor .

SECTION B

6. Creating Database Table:

```
53 • CREATE TABLE song(
54       song_id INTEGER NOT NULL DEFAULT 0,
55       song_name VARCHAR(30) NOT NULL UNIQUE,
56       song_language VARCHAR(10) NOT NULL,
57       duration TIME NOT NULL,
58       album_name VARCHAR(30) NOT NULL UNIQUE,
59       date_released DATE NOT NULL,
60       PRIMARY KEY(song_id),
61       FOREIGN KEY (album_name) REFERENCES album(album_name)
62   );
```

A Database table can be created using “CREATE TABLE” key word followed by the name we wish to give to the table. For example, the name of the table is song. From the second line, attributes for that table are initialized to create null tables with constraints. There are different datatypes that can be used in MySQL. Here, I used Integer for song_id, character (VARCHAR) for song_name with a maximum range of 30 characters allowed in the song name. In addition, for duration the datatype is TIME which takes in only a specific format (‘hh:mm:ss.00000’) and date_released, as previously mentioned, is of DATE Datatype and has format (‘YYYY-MM-DD’). The constraints taken for the song table can be given when the attribute is being initialized. For instance, all the attributes in the song table are designed such that they don’t have any null values while song_name and album_name also have unique constraints which is obeying the constraints/assumptions mentioned in the previous section. Primary Key of that table can be mentioned as given for this example and there can be only primary key. For indicating the foreign key in the table, references to the primary table and column must also be mentioned in the same way as shown. If there is more than one foreign key, the foreign keys can be listed down further in the same format of foreign key provided.

7. Altering Tables:

For this database, I did not have to alter tables using the alter command. But in general, if altering of a table is required, it can be done by specifying the table name we want to alter after the “ALTER TABLE” command. We can add a column to the table by using “ADD” key and the specify the column name with its definition.

```
87 •  ALTER TABLE play
88      ADD genre_name VARCHAR(20);
```

8. Trigger Operations:

These are automated events that occur behind when a database application is running. It can be update or insert either before or after the event occurring. For example, an addition of a new song before inserting, a trigger can be created using “CREATE TRIGGER” command followed by the trigger name by specifying if on every column or row, a new song_id is set for a new song_name in the song table.

```
CREATE TRIGGER  new_song_name
BEFORE INSERT
ON song
FOR EACH ROW
SET NEW.song_name = new.song_id;
```

9. Creating Views:

```
CREATE VIEW `Latest Album Releases` AS
SELECT album.date_released, album.album_name, album.no_of_songs, album.producer
FROM album
ORDER BY album.date_released DESC;


CREATE VIEW `Genre and Song` AS
SELECT singer.singer_name, singer.country_of_birth, singer.gender
FROM singer
WHERE singer.country_of_birth="United States";

```

Views are those which manipulate around the existing entities created already for the database application and creates a view based on the given conditions. A view can be created by using “CREATE VIEW” command followed by a name. The required columns for manipulation from the entities can be taken and must be mentioned from which table. The views are then created based on how and what data we select/retrieve from the entities. For example, here we took columns from album table and ordered it by its date released which gives us a view of latest albums in the database.

10. Populate Tables:

```
insert into singer values
   (1, "Ed Sheeran", "England", 18, "Male", '1991-02-17',31),
   (2, "Taylor Swift", "United States", 18, "Female", '1989-12-13',32),
   (3, "Ruby Carr", "South Africa", null, "Female", '1997-04-01',24),
   (4, "Natalia Panzarella", "United States", null, "Female", '1997-02-17',25),
   (5, "Olivia Isabel Rodrigo", "United States", 17, "Female", '2003-02-20',19),
   (6, "Shilpa Rao", "India", 15, "Female", '1984-04-11',37),
   (7, "Sreerama Chandra Mynampati", "India", 14, "Male", '1989-01-19',33),
   (8, "Luis Alfonso Rodríguez López-Cepero", "Puerto Rico", 24, "Male", '1978-04-15',43),
   (9, "Shawn Peter Raul Mendes", "Canada", 9, "Male", '1998-08-08',23),
   (10, "Ariana Grande-Butera", "United States", 14, "Female", '1993-06-26',28);
```

The tables can be populated by using “insert into” command and mention which table to insert into. It can be done by entering the values of each attribute in the same order and satisfying the character ranges and formats. “null” is entered when there is no information found.

11. Retrieving Information:

The information can be retrieved from the columns of tables. “SELECT” is a keyword which can be used to select the column names we wish to use in the trigger or anywhere. INNER JOIN will aid in joining the rows when the condition that song name from the song table must be equal to the album name from the album table is

met. There are other keywords as well like “ORDER BY” which helps to order the data either in ascending or descending. Distinct records of a table can be identified using “SELECT DISTINCT”. “AUTO INCREMENT” can be used as a function to generate primary keys as it goes down and we keep entering some details into the table.

```
CREATE VIEW `Album and Singer` AS
SELECT album.album_name, song.song_name, song.song_language
FROM song
INNER JOIN album
ON song.song_name=album.album_name;
```

12. Security Commands:

```
CREATE VIEW record_label_security AS
SELECT *
FROM record_label
WHERE distributor = "Universal Music Group";

GRANT SELECT ON record_label_security TO XXX WITH GRANT OPTION;
```

A security access or revoke can be granted to or taken away from users based on how the users are created. For that first, a view must be created as how mentioned previously and then security keyword “GRANT” can be used to grant access to user “XXX” on the view record_label_security based on the condition “WHERE”. In the same way, the access can be revoked from the user by using “REVOKE” keyword instead of “GRANT”. It doesn’t work for this database as there are no users assigned or in participation.

13. Other Features:

```
DELIMITER //
CREATE PROCEDURE get_song_names ()
BEGIN
SELECT song_name FROM song;
END//
DELIMITER //
```

A procedure can be created using delimiters at the start and end of the procedure. Between Begin and End, it can consist of all the SQL queries. There can be if clauses as well inside. It can be used to save some complex queries into a procedure and call them later when required simply.

```
show triggers in sql_project;
```

The SHOW TRIGGERS command in sql helps to see all the triggers in our database. Here my database is named as sql_project. In the same way, triggers can be dropped down by entering the trigger name after DROP TRIGGER.

```
drop trigger new_song_name;
```

14. CONCLUSION

The songs database has evident relationships and there are several SQL commands which can be made use of. The primary keys and foreign keys are mentioned accordingly for each entity.

SECTION C

15. Appendix

```
-- CREATE DATABASE sql_project;
USE sql_project;

CREATE TABLE instrument(
   instrument_name VARCHAR(30) NOT NULL,
   developed_in INTEGER NOT NULL DEFAULT 0,
   inventor VARCHAR(30) NOT NULL,
   classification VARCHAR(14) NOT NULL,
   PRIMARY KEY(instrument_name)
);

CREATE TABLE record_label(
   record_label_id VARCHAR(4) NOT NULL DEFAULT 0,
   record_name VARCHAR(30) NOT NULL UNIQUE,
   distributor VARCHAR(50) NOT NULL,
   country_origin VARCHAR(20) NOT NULL,
   founder VARCHAR(30) DEFAULT NULL,
   founded_in YEAR NOT NULL,
   PRIMARY KEY(record_label_id)
);

CREATE TABLE singer(
   singer_id INTEGER NOT NULL DEFAULT 0,
   singer_name VARCHAR(50) NOT NULL UNIQUE,
   country_of_birth VARCHAR(30) NOT NULL,
   years_active_till_present INTEGER DEFAULT NULL,
   gender VARCHAR(7) NOT NULL,
   date_of_birth DATE NOT NULL,
   age INTEGER NOT NULL DEFAULT 0,
```

```
   PRIMARY KEY(singer_id)
);

CREATE TABLE genre(
   genre_id INTEGER NOT NULL DEFAULT 0,
         genre VARCHAR(17) NOT NULL unique,
   PRIMARY KEY(genre_id)
);

CREATE TABLE album(
   album_id INTEGER NOT NULL DEFAULT 0,
   album_name VARCHAR(30) NOT NULL UNIQUE,
   no_of_songs INTEGER NOT NULL,
   length_of_album TIME NOT NULL,
   producer VARCHAR(30) DEFAULT NULL,
   singer_name VARCHAR(50) NOT NULL,
   record_label_id VARCHAR(4) NOT NULL,
   date_released DATE NOT NULL,
   PRIMARY KEY(album_id),
   FOREIGN KEY (singer_name) REFERENCES singer(singer_name),
   FOREIGN KEY (record_label_id) REFERENCES record_label(record_label_id)
);

CREATE TABLE song(
   song_id INTEGER NOT NULL DEFAULT 0,
   song_name VARCHAR(30) NOT NULL UNIQUE,
   song_language VARCHAR(10) NOT NULL,
   duration TIME NOT NULL,
   album_name VARCHAR(30) NOT NULL UNIQUE,
   date_released DATE NOT NULL,
   PRIMARY KEY(song_id),
   FOREIGN KEY (album_name) REFERENCES album(album_name)
);

CREATE TABLE perform(
   song_name VARCHAR(30) NOT NULL,
   singer_name VARCHAR(50) NOT NULL,
   FOREIGN KEY (song_name) REFERENCES song(song_name),
   FOREIGN KEY (singer_name) REFERENCES singer(singer_name)
);

CREATE TABLE belong(
   song_id INTEGER NOT NULL DEFAULT 0,
   genre_id INTEGER NOT NULL DEFAULT 0,
   FOREIGN KEY (song_id) REFERENCES song(song_id),
   FOREIGN KEY (genre_id) REFERENCES genre(genre_id)
);
```

```
CREATE TABLE play(
   song_id INTEGER NOT NULL DEFAULT 0,
         instrument_name VARCHAR(30) NOT NULL,
   FOREIGN KEY (song_id) REFERENCES song(song_id),
   FOREIGN KEY (instrument_name) REFERENCES instrument(instrument_name)
);



ALTER TABLE play
  ADD genre_name VARCHAR(20);


insert into instrument values
   ("Acoustic Guitar", 1796, "Christian Frederick Martin", "String"),
   ("Piano", 1700, "Bartolomeo Cristofori", "Keyboard"),
   ("Programmed Drums", 1957, "Harry Chamberlin", "Drum"),
   ("Banjo", 1700, "Joel Walker Sweeney", "chordophone "),
   ("Flute", 0900, "Theobald Boehm", "Woodwind"),
   ("Violin", 1500, "Andrea Amati", "chordophone"),
   ("Oboe", 1650, "Jacques Hotteterre", "Wind"),
   ("Trumpet", 1922, "Howard Carter", "Brass"),
   ("Soprano Saxophone", 1846, "Adolphe Sax", "Woodwind");

insert into record_label values
   ("AS12", "Asylum", "Warner Music Group", "United States", "David Geffen", 1971),
   ("BM05", "Big Machine", "Universal Music Group", "United States", "Scott Borchetta",
2004),
   ("RP34", "Republic", "Universal Music Group", "United States", "Monte Lipman", 1995),
   ("G09", "Geffen", "Interscope Geffen A&M", "United States", "David Geffen", 1980),
   ("TS78", "T-Series", "Music record label", "India", "Gulshan Kumar", 1983),
   ("UL11", "Universal Latin", "Universal Music Group", "United States", null , 2008),
   ("I55", "Island", "EMI records", "United Kingdom", "Chris Blackwell", 1959);

 insert into singer values
   (1, "Ed Sheeran", "England", 18, "Male", '1991-02-17',31),
   (2, "Taylor Swift", "United States", 18, "Female", '1989-12-13',32),
   (3, "Ruby Carr", "South Africa", null, "Female", '1997-04-01',24),
   (4, "Natalia Panzarella", "United States", null, "Female", '1997-02-17',25),
   (5, "Olivia Isabel Rodrigo", "United States", 17, "Female", '2003-02-20',19),
   (6, "Shilpa Rao", "India", 15, "Female", '1984-04-11',37),
   (7, "Sreerama Chandra Mynampati", "India", 14, "Male", '1989-01-19',33),
   (8, "Luis Alfonso Rodríguez López-Cepero", "Puerto Rico", 24, "Male", '1978-04-15',43),
   (9, "Shawn Peter Raul Mendes", "Canada", 9, "Male", '1998-08-08',23),
   (10, "Ariana Grande-Butera", "United States", 14, "Female", '1993-06-26',28);
```

```
insert into genre values
   (186, "Folk Pop"),
   (243, "Hip hop"),
   (873, "Country"),
   (342, "Soft rock"),
   (456, "R&B"),
   (372, "Reggaeton"),
   (546, "trap"),
   (83, "Pop"),
   (324, "Latin pop"),
   (765, "Electro pop");

insert into album values
   (11, "x", 5, '00:50:23.000000',"Benny Blanco","Ed Sheeran", "AS12",'2014-06-20'),
   (22, "Red", 7, '01:05:09.000000',"Taylor Swift","Taylor Swift", "BM05",'2012-10-22'),
   (33, "Thank U, Next", 3, '00:41:11.000000',"Charles Anderson","Ariana Grande-Butera",
"RP34",'2019-02-08'),
   (44, "÷", 5, '00:46:14.000000',"Benny Blanco","Ed Sheeran", "AS12",'2017-03-03'),
   (55, "Sour", 5, '00:34:41.000000',"Dan Nigro","Olivia Isabel Rodrigo", "G09",'2021-05-21'),
   (66, "Yeh Jawaani Hai Deewani", 10, '00:40:19.000000',"Karan Johar","Sreerama Chandra
Mynampati", "TS78",'2013-04-29'),
   (77, "Vida", 5, '00:52:25.000000',"Mauricio Rengifo","Luis Alfonso Rodríguez López-
Cepero", "UL11",'2019-02-01'),
   (88, "1989", 7, '00:48:41.000000',"Max Martin","Taylor Swift", "BM05",'2014-10-27'),
   (99, "illuminate", 3, '00:44:34.000000',"Jake Gosling","Shawn Peter Raul Mendes",
"I55",'2016-09-23');

insert into song values
   (101, "Photograph", "English", '00:04:19.000000',"x",'2015-05-11'),
   (202, "Red", "English", '00:03:40.000000',"Red",'2013-06-24'),
   (303, "7 Rings", "English", '00:02:58.000000',"Thank U, Next",'2019-01-18'),
   (404, "Perfect", "English", '00:04:23.000000',"÷",'2017-09-26'),
   (505, "Drivers License", "English", '00:04:02.000000',"Sour",'2021-01-08'),
   (606, "Subanallah", "Hindi", '00:04:11.000000',"Yeh Jawaani Hai Deewani",'2013-05-31'),
   (707, "Despacito", "Spanish", '00:03:47.000000',"Vida",'2017-01-12'),
   (808, "Blank Space", "English", '00:03:52.000000',"1989",'2014-11-10'),
   (909, "Treat you better", "English", '00:03:07.000000',"Illuminate",'2016-06-03');

insert into perform values
   ("Photograph","Ed Sheeran"),
   ("Red","Taylor Swift"),
   ("7 Rings","Ariana Grande-Butera"),
   ("Perfect","Ed Sheeran"),
   ("Drivers License","Olivia Isabel Rodrigo"),
   ("Subanallah","Shilpa Rao"),
   ("Subanallah","Sreerama Chandra Mynampati"),
   ("Despacito","Luis Alfonso Rodríguez López-Cepero"),
```

```
   ("Blank Space","Taylor Swift"),
   ("Treat You Better","Shawn Peter Raul Mendes");

insert into belong values
   (101, 186),
   (202, 873),
   (202, 342),
   (303, 546),
   (303, 243),
   (303, 456),
   (404, 83),
   (707, 324),
   (707, 372),
   (808, 765);

insert into play values
   (101, "Acoustic Guitar"),
   (101, "Piano"),
   (101, "Programmed Drums"),
   (202, "Banjo"),
   (202, "Acoustic Guitar"),
   (404, "Flute"),
   (404, "Violin"),
   (404, "Oboe");

CREATE VIEW `Latest Album Releases` AS
SELECT album.date_released, album.album_name, album.no_of_songs, album.producer
FROM album
ORDER BY album.date_released DESC;


CREATE VIEW `Genre and Song` AS
SELECT singer.singer_name, singer.country_of_birth, singer.gender
FROM singer
WHERE singer.country_of_birth="United States";

CREATE VIEW `Album and Singer` AS
SELECT album.album_name, song.song_name, song.song_language
FROM song
INNER JOIN album
ON song.song_name=album.album_name;

CREATE VIEW record_label_security AS
SELECT *
FROM record_label
WHERE distributor = "Universal Music Group";
GRANT SELECT ON record_label_security TO XXX WITH GRANT OPTION;
```

```
CREATE TRIGGER  new_song_name
BEFORE INSERT
ON song
FOR EACH ROW
SET NEW.song_name = new.song_id;

DELIMITER //
CREATE PROCEDURE get_song_names ()
BEGIN
SELECT song_name FROM song;
END//
DELIMITER //

show triggers in sql_project;
drop trigger new_song_name;
```