MUSIC APP DATABASE MANAGEMENT SYSTEM

Komal Yadav (2019IMT-050)

Problem Statement

A music app is planning to implement a database to enhance its data management practice and ultimately advance its business operations.

The initial planning analysis phases have revealed the following system requirements:

- Each album has a unique Album ID as well as the following attributes: Album Title, Year. An album contains at least one song or more songs.
- Songs are identified by Song ID.
- Each song can be contained in more than one album or not contained in any of them at all and has a Song Title, Year and Duration.
- Each song belongs to at least one genre or multiple genres.

- Songs are written by at least an artist or multiple artists.
- Each artist has a unique Artist ID corresponding to his or her name, and an artist writes at least one song or multiple songs, to be recorded in the database or releases one or multiple Albums on his own or in collaboration with different artists.
- Also many artists can collaborate to write a song together which may or may not belong to their solo albums.
- To use the app, one needs to first sign up on the app as a user giving details of his name, date of birth, phone number.
- Each user is identified by a unique User ID.

- Also the user can subscribe to a premium version of the app to get rid off the advertisements on the app and enjoy features like offline playback.
- To subscribe to a premium version of the app the user has to select a plan. Each Premium Plan has its own unique ID corresponding to the duration of the plan and price.
- The details such as method of payment and date of subscription are also maintained.
- Also the user account has other attributes like Liked Songs, Liked Albums, Playlists and Frequents.

NORMALIZED TABLES

First Normal Form: First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.

Each attribute must contain only a single value from its pre-defined domain.

For example: We have two values in SongName , ‘Normal’ and ‘Lucky you’. So you would make a new tuple(row) dividing the previous one.

So it would be (SongName , ‘Normal’) and (SongName , ‘Lucky you’)

Second Normal Form: It uses the concept of **Prime attribute** and **Non-prime attribute.**If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if $X \rightarrow A$ holds, then there should not be any proper subset Y of X, for which $Y \rightarrow A$ also holds true.

Third Normal Form: For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

- No non-prime attribute is transitively dependent on prime key attribute.
- For any non-trivial functional dependency, X → A, then either −
    - X is a superkey or,
    - A is prime attribute.

Boyce-Codd Normal Form: Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

- For any non-trivial functional dependency, X → A, X must be a super-key.

ENTITY RELATIONSHIP DIAGRAM

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

It consists of Entities, their Attributes an the Relationship between the Entities.

Entity : An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. In our project we took Album, Song, Genre, Playlist, Premium Plan, User and Artist as entities. They are represented by rectangular blocks.

Attributes: : Entities are represented by means of their properties, called **attributes**. All attributes have values.

Keys : Key is an attribute or collection of attributes that uniquely identifies an entity among entity set. We have different types of Keys which are:

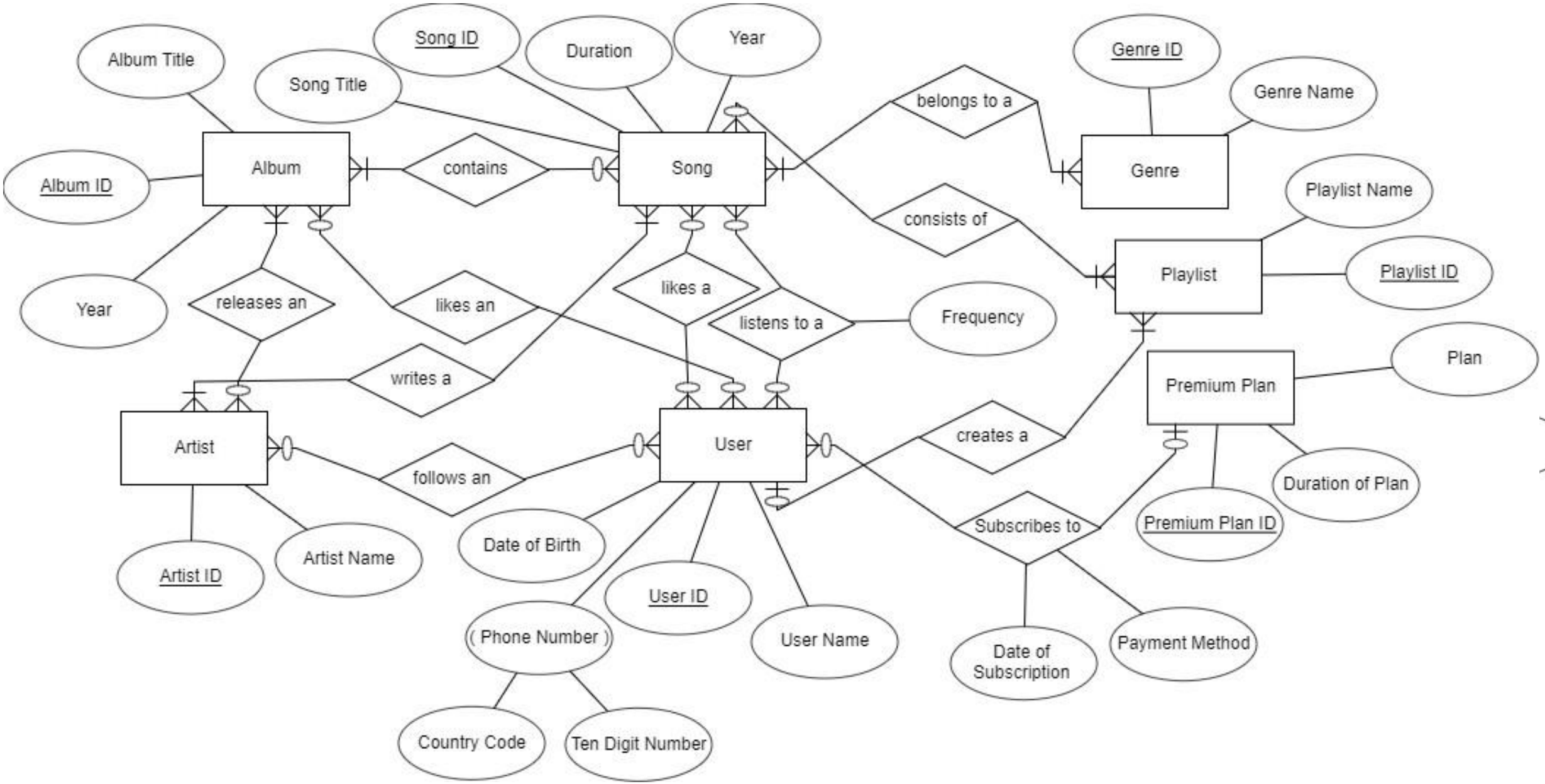
- Super Key
- Candidate Key
- Primary Key

In the ER model, the attributes that are Keys are underlined.

Relationship : The association among entities is called a relationship.

Relationships can also be categorized based on it’s degree as:

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree


Album Title
Song ID
Duration
Year
Song Title
Genre ID
Genre Name
belongs to a
Album
contains
Song
Genre
Album ID
Playlist Name
consists of
Playlist
Playlist ID
Year
releases an
likes an
likes a
listens to a
Frequency
writes a
Plan
Premium Plan
Artist
follows an
User
creates a
Duration of Plan
Subscribes to
Premium Plan ID
Artist ID
Artist Name
Date of Birth
User ID
(Phone Number)
User Name
Date of Subscription
Payment Method
Country Code
Ten Digit Number

Database Design

Entities And Attributes

1. Album
   - Album Id
   - Album Title
   - Year

2. Artist
   - Artist Id
   - Artist name

3. User
   - User Id
   - Username
   - Date of birth
   - Phone number

4. Song
   - Song Id
   - Duration
   - Song title
   - Year

5. Genre
   - Genre Id
   - Genre name

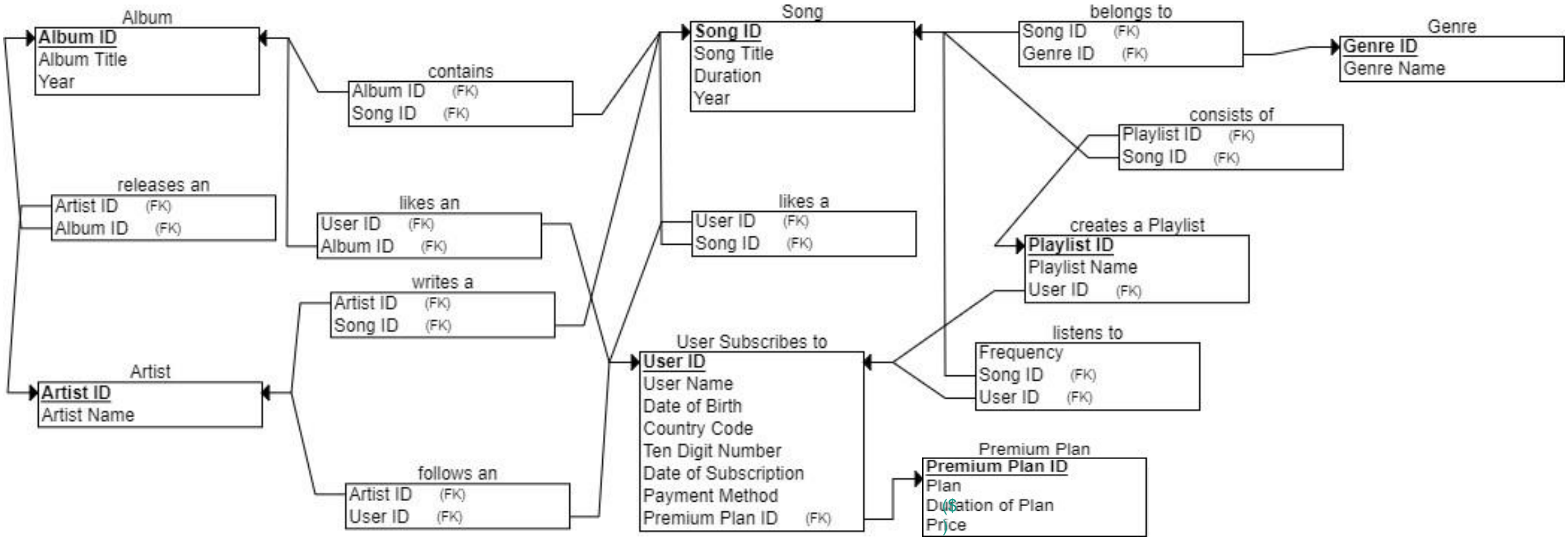
6. Playlist
   - Playlist Id
   - Playlist name

7. Premium plan
   - Premium plan id
   - Duration of plan
   - Price
   - Plan

Relations

1. Contains
   - Album contains song
   - Many to many
1. Likes
   - User likes an album
   - Many to many
2. Releases
   - Artist releases an album
   - many to many
3. Writes
   - Artist writes a song
   - many to many
4. Follows
   - User follows an artist
   - Many to many
5. Listens
   - User listens to a song
   - Many to many
6. Subscribe to
   - User subscribes to premium plan
   - many to one
7. Creates
   - User creates a playlist
   - many to many
8. Consists of
   - Playlist consists a song
   - many to many
9. Belongs to
   - Song belongs to a genre
   - many to many

Conversion Of ER Diagram To Relational Schema


Album
Album ID
Album Title
Year
contains
Album ID (FK)
Song ID (FK)
Song
Song ID
Song Title
Duration
Year
belongs to
Song ID (FK)
Genre ID (FK)
Genre
Genre ID
Genre Name
consists of
Playlist ID (FK)
Song ID (FK)
releases an
Artist ID (FK)
Album ID (FK)
likes an
User ID (FK)
Album ID (FK)
likes a
User ID (FK)
Song ID (FK)
creates a Playlist
Playlist ID
Playlist Name
User ID (FK)
writes a
Artist ID (FK)
Song ID (FK)
User Subscribes to
User ID
User Name
Date of Birth
Country Code
Ten Digit Number
Date of Subscription
Payment Method
Premium Plan ID (FK)
listens to
Frequency
Song ID (FK)
User ID (FK)
Artist
Artist ID
Artist Name
follows an
Artist ID (FK)
User ID (FK)
Premium Plan
Premium Plan ID
Plan
Duration of Plan
Price

Belongs to

Song ID	Genre ID
1	1
1	2
2	1
2	2
3	1
3	2
4	3
4	5
5	3
6	3
7	1
7	2
8	1
8	2
9	1
9	2
10	5
11	5
12	4
13	6
14	3
15	1
15	2
15	4
16	3
17	5

Song

Song ID	Song Title	Duration	Year
1	The Ringer	00:05:38	2018
2	Normal	00:03:43	2018
3	Lucky You	00:04:05	2018
4	Some Nights	00:04:37	2012
5	We Are Young	00:04:10	2012
6	Why Am I the One	00:04:46	2012
7	Rap God	00:06:03	2013
8	The Monster	00:04:10	2013
9	Headlights	00:05:43	2013
10	American Idiot	00:02:54	2004
11	Wake me up when September Ends	00:07:13	2004
12	Holiday/Boulevard of Broken Dreams	00:08:13	2004
13	Alone	00:03:20	2016
14	Viva La Vida	00:05:19	2008
15	Heathens	00:03:15	2016
16	Complicated	00:03:04	2017
17	bad guy	00:03:14	2019

Genre

Genre ID	Genre Name
1	Hip-Hop
2	Rap
3	Pop
4	Rock
5	Indie
6	Dance/Electronic

Artist

Artist ID	Artist Name
1	Eminem
2	Joyner Lucas
3	fun.
4	Janelle Monáe
5	Rihanna
6	Nate Ruess
7	Green Day
8	Marshmello
9	Coldplay
10	Twenty One Pilots
11	David Guetta
12	Dimitri Vegas
13	Like Mike
14	Billie Eilish

Premium Plan

Premium Plan ID	Plan	Duration of Plan	Price($)
1	Value Pack	1	5
2	Value Pack	3	10
3	Value Pack	6	15
4	Value Pack	9	20
5	Value Pack	12	25

Contains

Album ID	Song ID
1	1
1	2
1	3
2	4
2	5
2	6
3	7
3	8
3	9
4	10
4	11
4	12

Album

Album ID	Album Title	Year
1	Kamikaze	2018
2	Some Nights	2012
3	The Marshall Mathers LP2	2013
4	American Idiot	2004

Releases an

Artist ID	Album ID
1	1
3	2
1	3
7	4

Writes a

Artist ID	Song ID
1	1
1	2
1	3
2	3
3	4
3	5
4	5
3	6
1	7
1	8
5	8
1	9
6	9
7	10
7	11
7	12
8	13
9	14
10	15
11	16
12	16
13	16
14	17

User Subscribes to

User ID	User Name	Date of Birth	Country Code	Ten Digit Number	Date of Subscription	Payment Method	Premium Plan ID
1	Aryan Chauhan	2001-08-04	+91	8836885789	2020-05-13	UPI	1
2	Dhruv Datta	2001-10-30	+91	9818896547	null	null	null
3	Aryan Anand	2001-06-24	+44	8368975098	null	null	null
4	Arjun Sharma	2001-09-05	+91	9868723161	2020-05-15	Paytm	3

Follows an

User ID	Artist ID
1	1
1	9
1	11
2	9
2	11
4	1
4	3
4	9
4	11
4	8

Likes an

User ID	Album ID
1	1
1	3
2	2
2	4
3	4
4	1
4	2

Likes a

User ID	Song ID
1	7
1	8
1	12
1	13
2	2
2	9
3	14
3	16
4	4
4	5
4	7

Listens to

User ID	Song ID	Frequency
1	4	7
1	7	14
1	8	29
1	9	1
1	10	24
1	11	28
2	1	3
2	2	16
2	3	25
2	7	19
2	8	25
2	9	12
3	4	30
3	5	34
3	6	37
3	14	20
3	17	10
4	7	11
4	8	16
4	9	2
4	13	21
4	15	33

Creates a Playlist

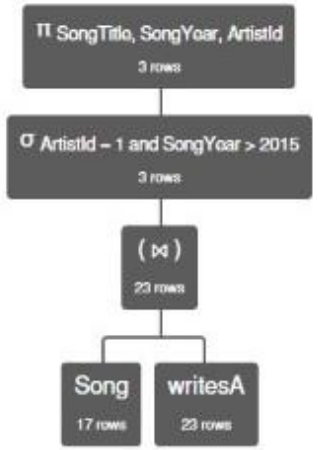
Playlist ID	Playlist Name	User ID
1	BANGERS	1
2	Rap	1
3	Rock Classics	2
4	EDM	3
5	2010s	4

Consists of

Playlist ID	Song ID
1	8
1	9
1	14
1	15
1	16
2	1
2	2
2	3
2	7
3	10
3	11
3	12
4	13
4	16
5	4
5	5
5	6
5	14

SOME BASIC RELAX QUERIES:

Q1-Write a query to find the songs released after the year 2015 by the artist having artist id=1 ?

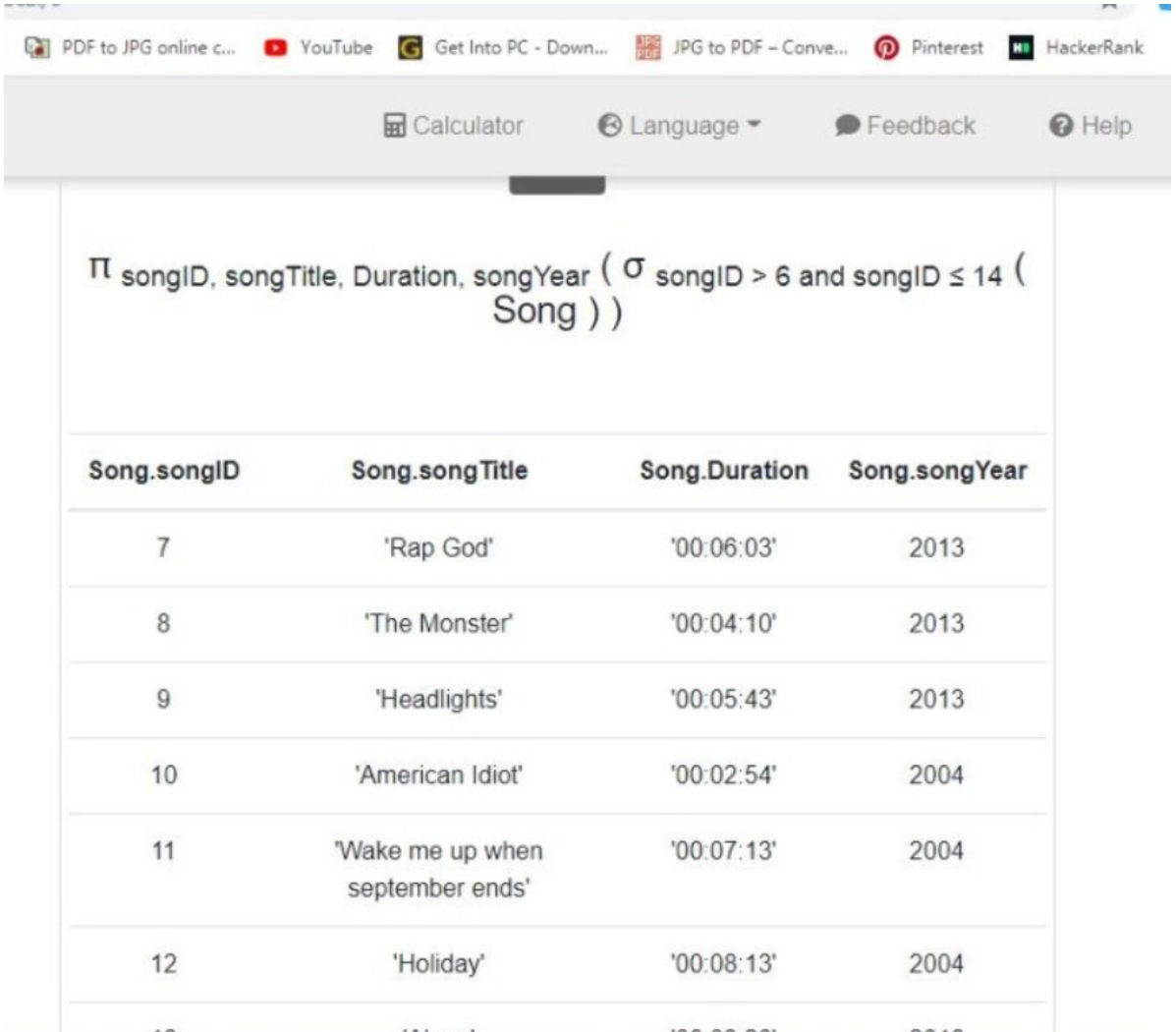


$$\pi_{\text{SongTitle, SongYear, ArtistId}} (\sigma_{\text{ArtistId = 1 and SongYear > 2015}} (\text{Song} \bowtie \text{writesA}))$$

Song.SongTitle	Song.SongYear	writesA.ArtistId
'The Ringer'	2018	1
'Normal'	2018	1
'Lucky You'	2018	1

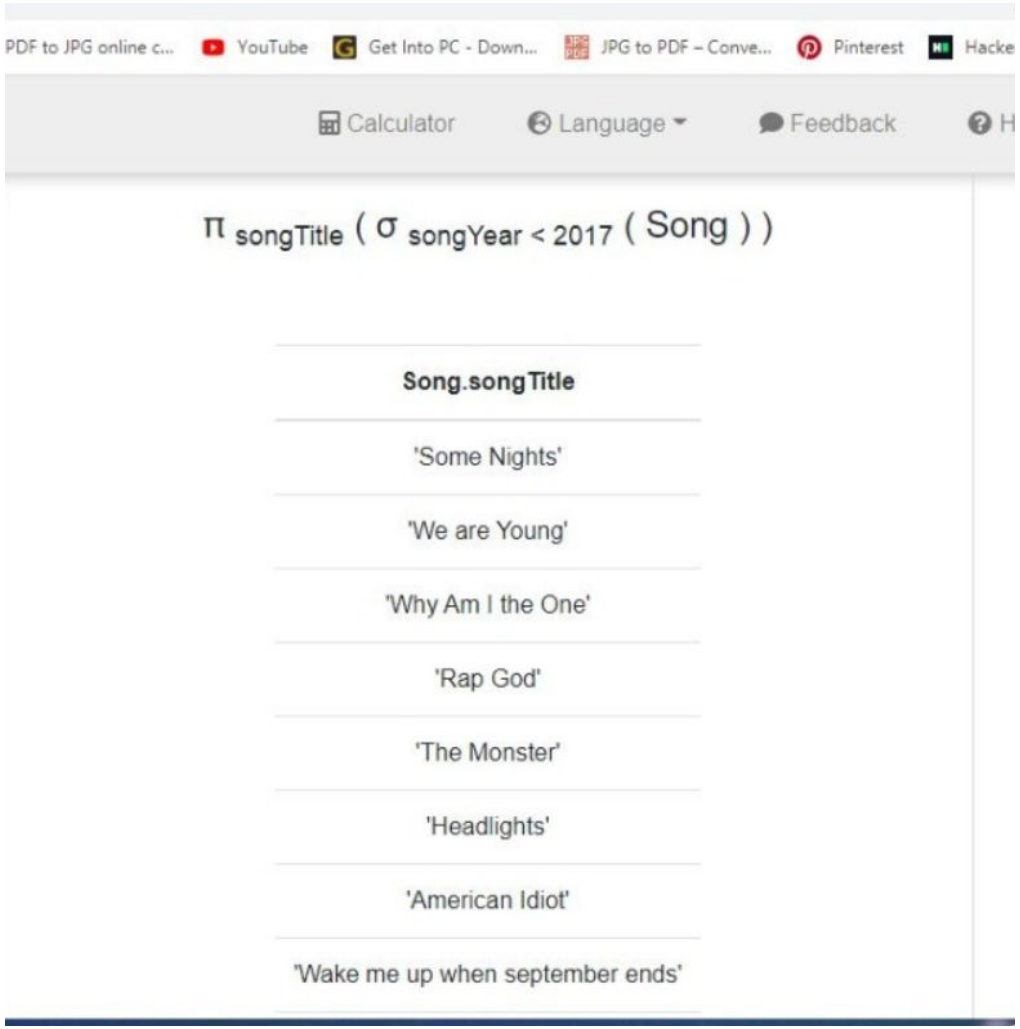
‹ 1 ›

Q2-Write A Query to display SongID, SongTitle, Duration and Song Year which have SongID greater than 6 and SongID less than 14.

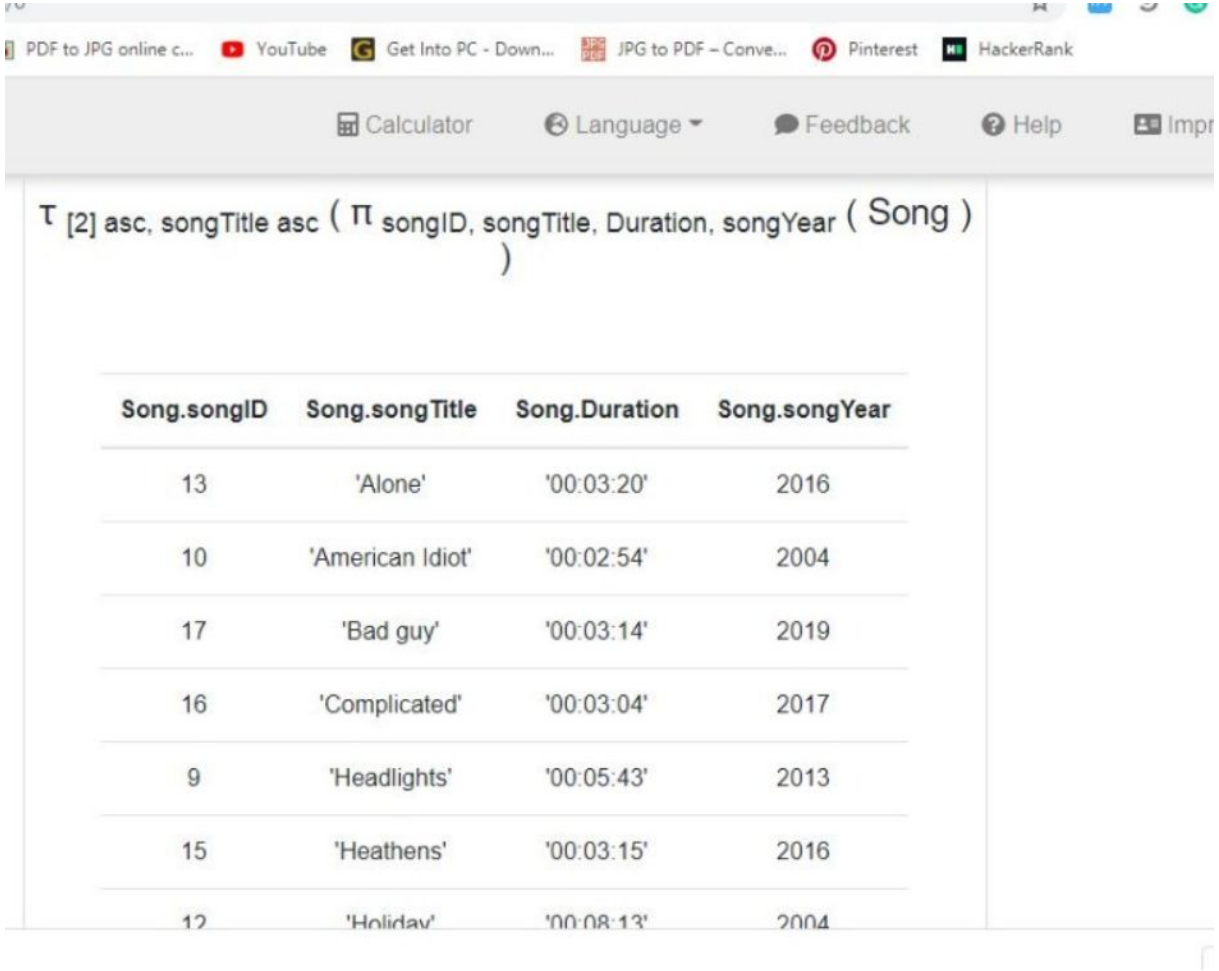


Song.songID	Song.songTitle	Song.Duration	Song.songYear
7	'Rap God'	'00:06:03'	2013
8	'The Monster'	'00:04:10'	2013
9	'Headlights'	'00:05:43'	2013
10	'American Idiot'	'00:02:54'	2004
11	'Wake me up when september ends'	'00:07:13'	2004
12	'Holiday'	'00:08:13'	2004

Q3-Write a query to display all song titles before year 2017 ?



Q4-Query to display song ID ,Duration , Song Year and Song Title In Ascending order



Song.songID	Song.songTitle	Song.Duration	Song.songYear
13	'Alone'	'00:03:20'	2016
10	'American Idiot'	'00:02:54'	2004
17	'Bad guy'	'00:03:14'	2019
16	'Complicated'	'00:03:04'	2017
9	'Headlights'	'00:05:43'	2013
15	'Heathens'	'00:03:15'	2016
12	'Holiday'	'00:08:13'	2004

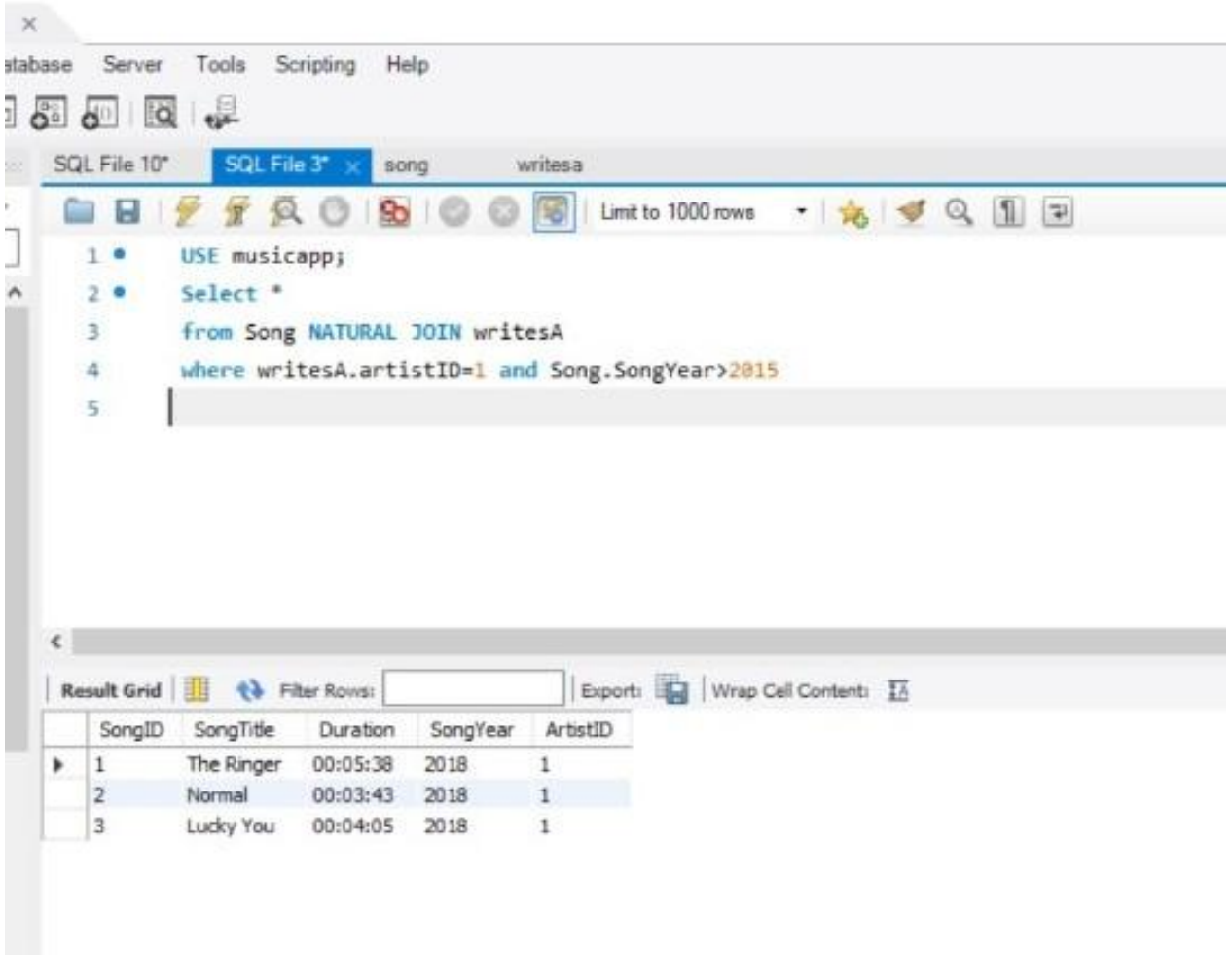
Q5-Write a query to display User liked songs with genre name for user having id=1 ?

$$\pi_{\text{SongTitle, GenreName}} (\sigma_{\text{UserID} = 1} (((\text{LikesA} \bowtie \text{Song}) \bowtie \text{BelongsTo}) \bowtie \text{Genre}))$$

Song.SongTitle	Genre.GenreName
'Rap God'	'Hip-Hop'
'Rap God'	'Rap'
'The Monster'	'Hip-Hop'
'The Monster'	'Rap'
'Holiday'	'Rock'
'Alone'	'Dance/Electronic'

SQL Queries:

Q1-Write a query to find the songs released after the year 2015 by the artist having artist id=1 ?



Q2-Write a query to display all the songs and their genre liked by user having user id=1 .

Query->

```
1   USE musicapp;
2   SELECT SongTitle,GenreName
3   from likesa natural join song natural join belongsto natural join genre
4   WHERE likesa.UserID = 1
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SongTitle	GenreName
Rap God	Hip-Hop
Rap God	Rap
The Monster	Hip-Hop
The Monster	Rap
Holiday/Boulevard of Broken Dreams	Rock
Alone	Dance/Electronic

Q3-Write a query to display the number of songs released in the year=2012.

Query

```
234 •   SELECT COUNT(SongYear)
235     FROM Song
236     WHERE SongYear='2012';
```

Result Grid | Filter

	COUNT(SongYear)
▶	3

Q4-Write a query to display song id,title,duration and year of all songs having song id greater than 6 and less than equal to 14.

Query

Song.songID	Song.songTitle	Song.Duration	Song.songYear
7	'Rap God'	'00:06:03'	2013
8	'The Monster'	'00:04:10'	2013
9	'Headlights'	'00:05:43'	2013
10	'American Idiot'	'00:02:54'	2004
11	'Wake me up when september ends'	'00:07:13'	2004
12	'Holiday'	'00:08:13'	2004
13	'Alone'	'00:03:20'	2016

Q5-Write a query to display all users that have suscribed to premium plan worth 15$ or more ?

Query

```
1 •  USE musicapp;
2 •  SELECT UserName,price$
3    FROM usersubscribesto NATURAL JOIN premiumplan
4    WHERE premiumplan.price$>=15
5
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

UserName	price$
Arjun Sharma	15

CONCLUSION

We first started by defining our problem. Almost every person listens to music and over the centuries there have been many songs and albums. So this clearly tells us that there must be a lot of data and we would eventually need a database to store and manage all of it.

Then we listed out all the entities and their attributes, formed relations between them and made the ER diagram. It was a little hard to make it at first but we could make it after referring to a few websites and video lectures.

Then we proceeded to convert the ER diagram to the relational schema. This was fairly easy.

To normalize the tables and get the functional dependencies we followed the manual method. Doing so we found that the database became more accurate and reduced the storage space.

Throughout the project we referred to various websites and the resources our professor Dr. Debanjan Sandhya shared with us. We could understand the subject better doing this project hands-on and could learn how to apply SQL as well.

Thanking You !