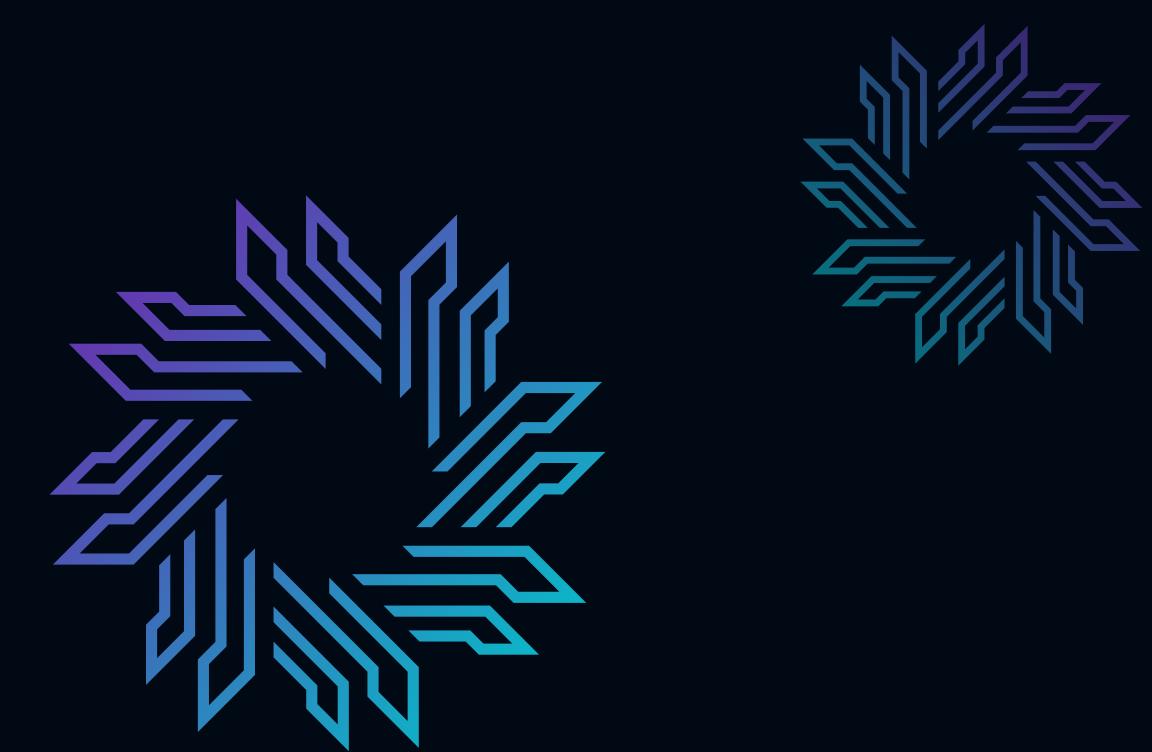


ONLINE MUSIC APPLICATION DATABASE

—● Submitted to: Dr. Debanjan Sadhya ●—



Submitted by:

Aleesha (2021BCS - 011)

Himanshi (2021BCS - 032)

Mishika Jaiswal (2021BCS - 043)

Roshan Jaiswal (2021BCS - 79)

Sakshi (2021BCS - 059)

Utkarsh Mishra (2021BCS - 68)

Contents

1. Overview

A brief introduction to our project

2. Significance and usefulness

3. ER Diagram

4. Entities and Attributes

5. Entity sets to tables

6. Relationship sets and cardinality

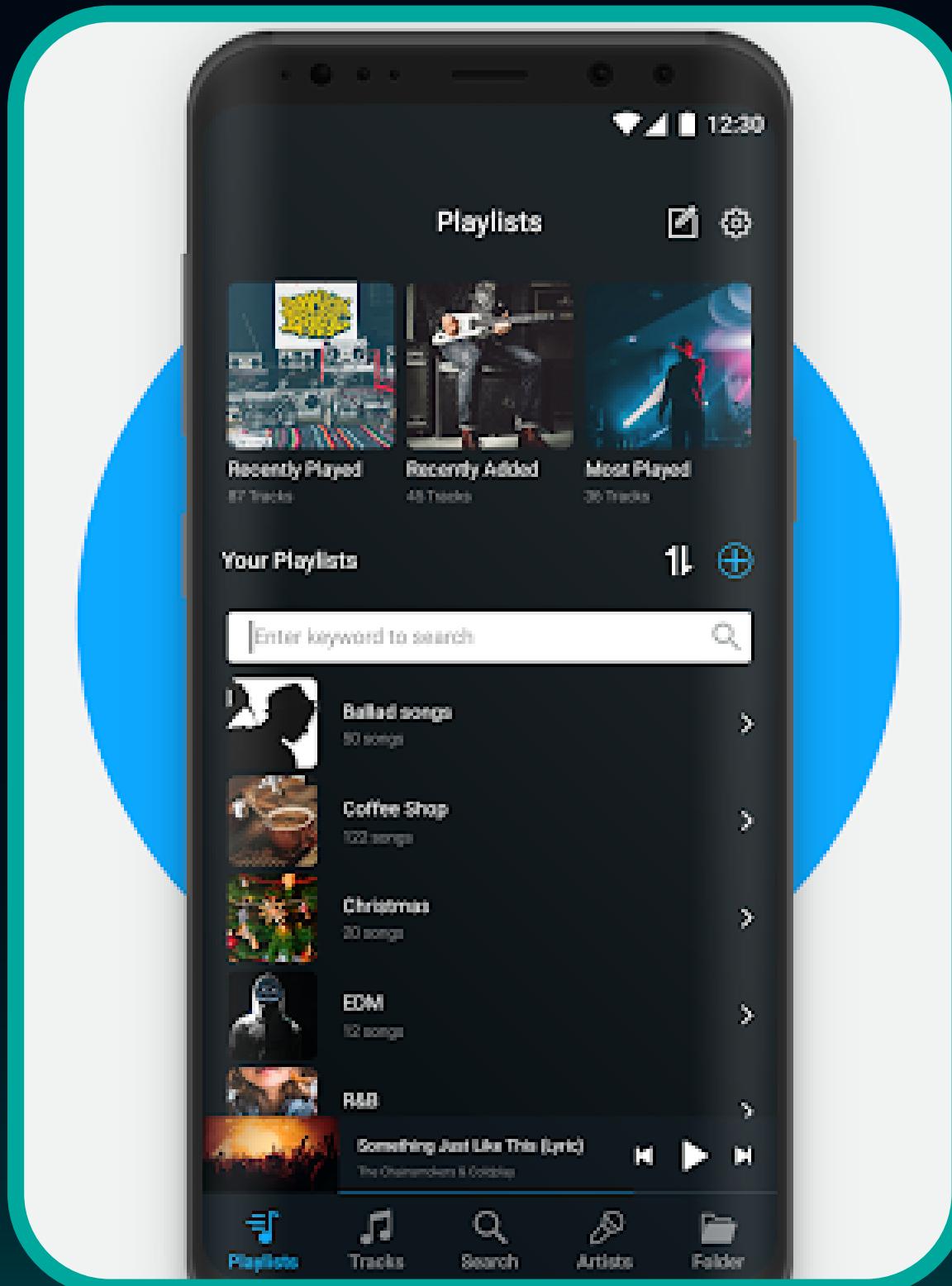
6. SQL

7. Normalisation

9. Problems in SQL

10. Problems in Relational Algebra

Overview



"The best music is essentially there to provide you something to face the world with."

Everyone has their own personal tastes and preferences when it comes to choosing the right music.

Therefore, here we bring our database of the online streaming music application, so that it's easy to find the right music for every moment on your phone, your computer, your tablet and more. So whether you're behind the wheel, working out, or relaxing, the right music is always at your fingertips.

Choose what you want to listen to.

Usefulness

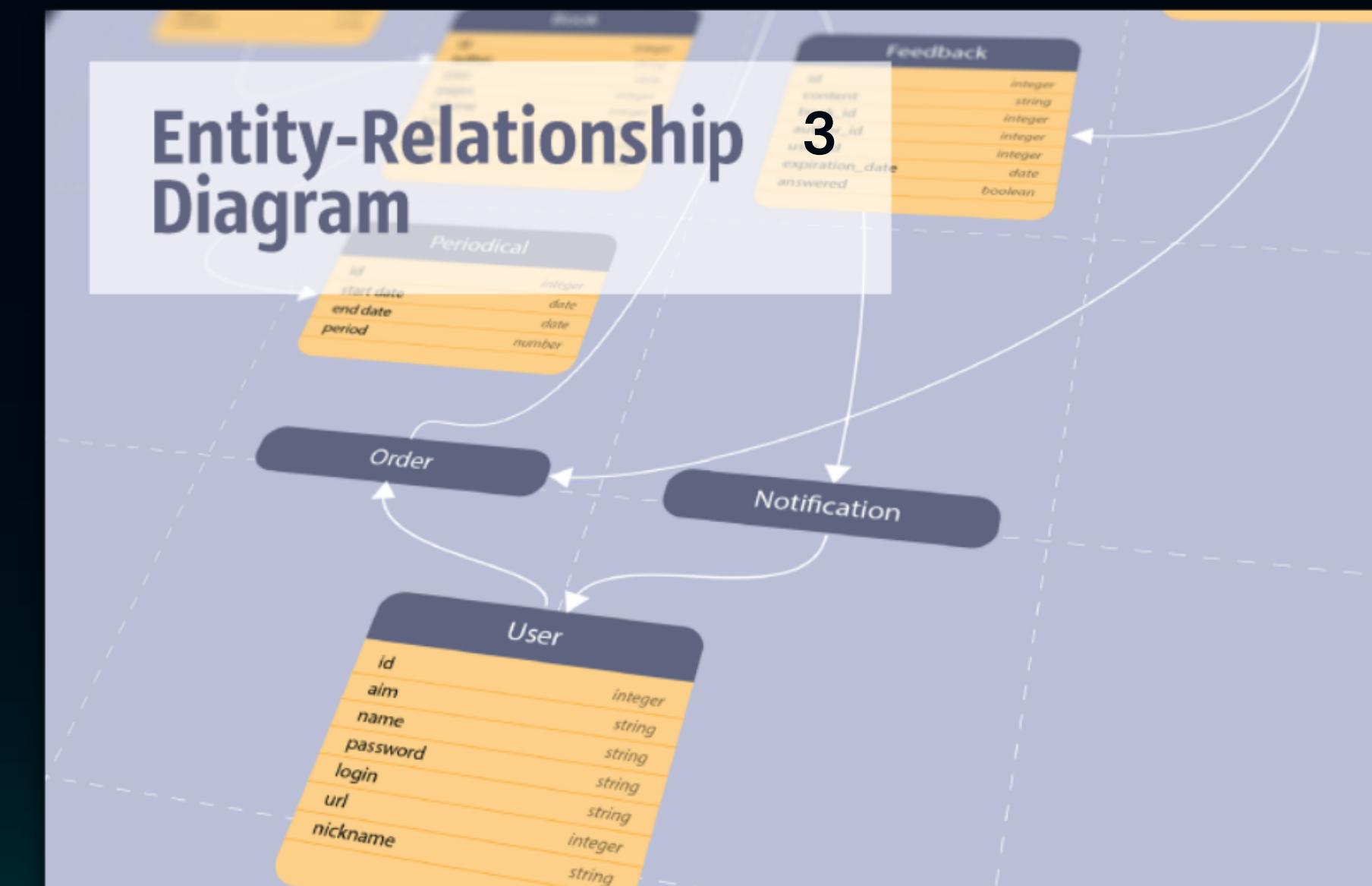
- ① It will help the users to get an exposure to a vast variety of music.
- ② It will Facilitate Easy Distribution of Music. The streaming platforms allow artists to distribute their new songs, albums, remixes, etc. through different channels in the network.
- ③ Accessing songs according to one's tastes would be easier through this database.
- ④ It is user-friendly as it provides various ways in which one can make the payment.
- ⑤ You don't need to subscribe to multiple channels.

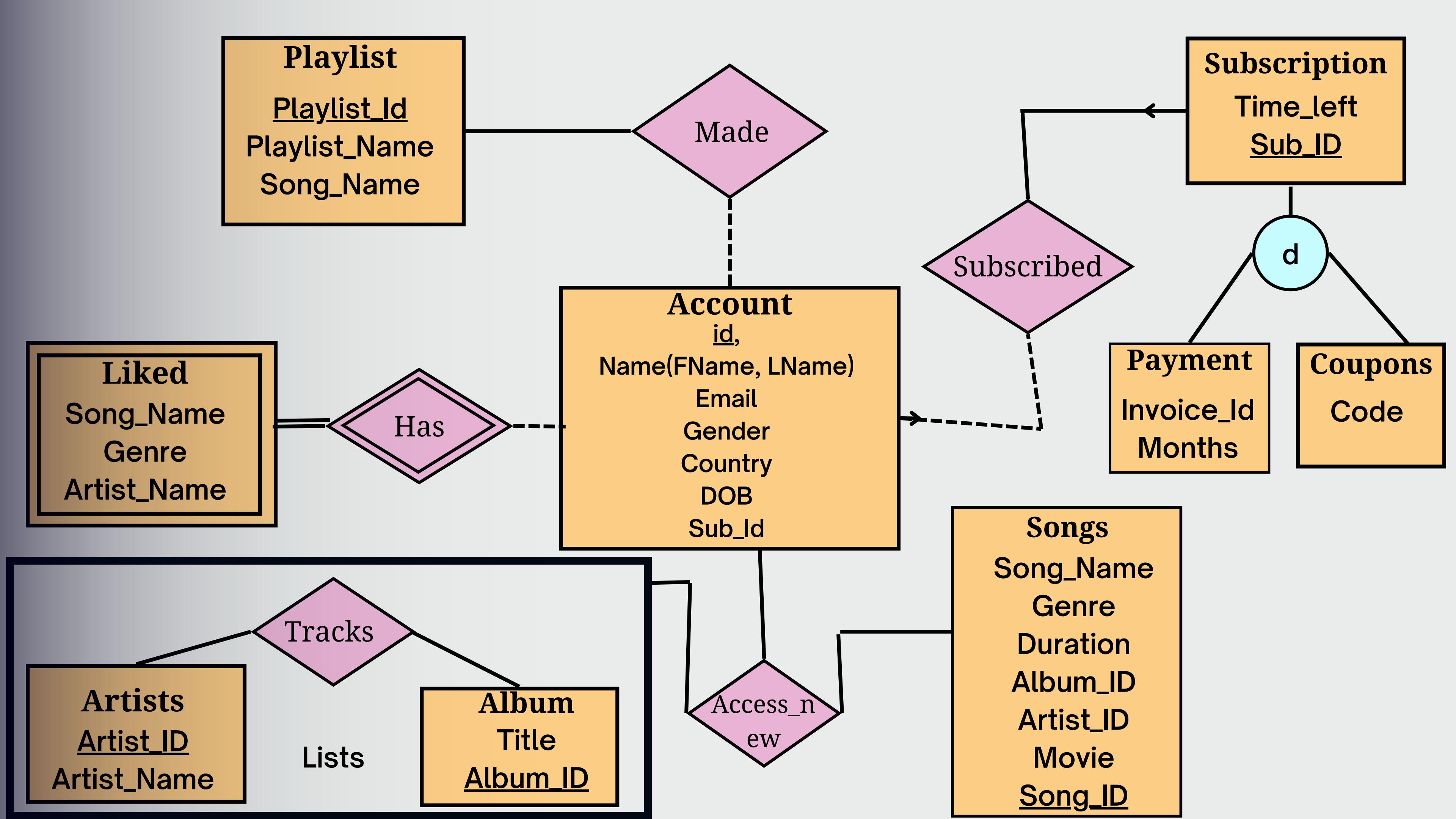


ER DIAGRAM

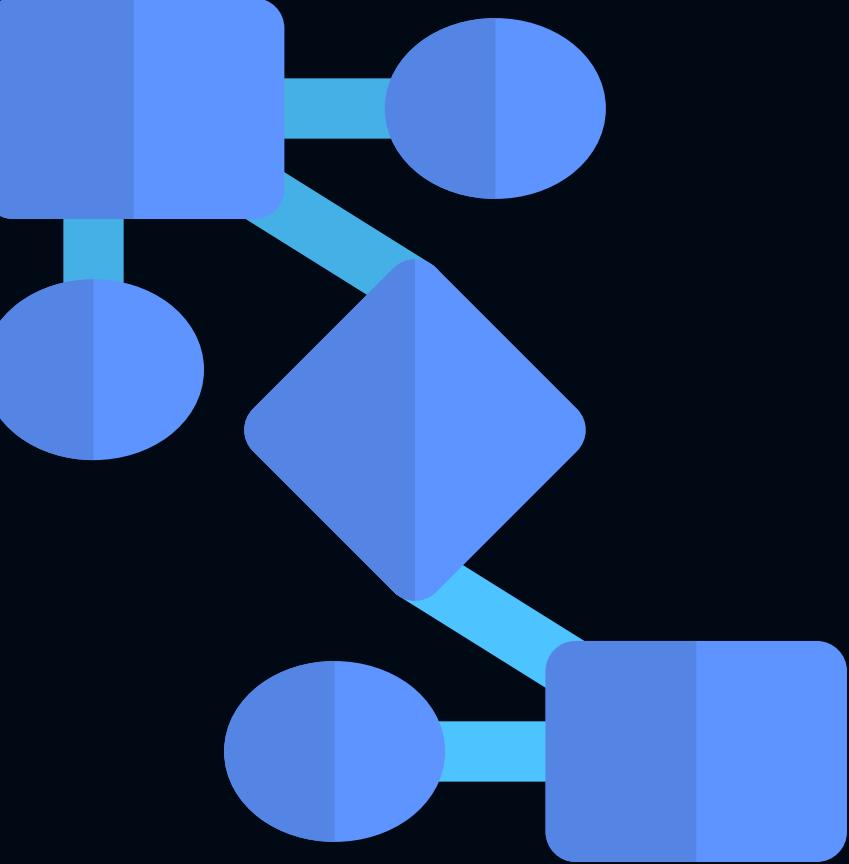
The ER data model was developed to facilitate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.

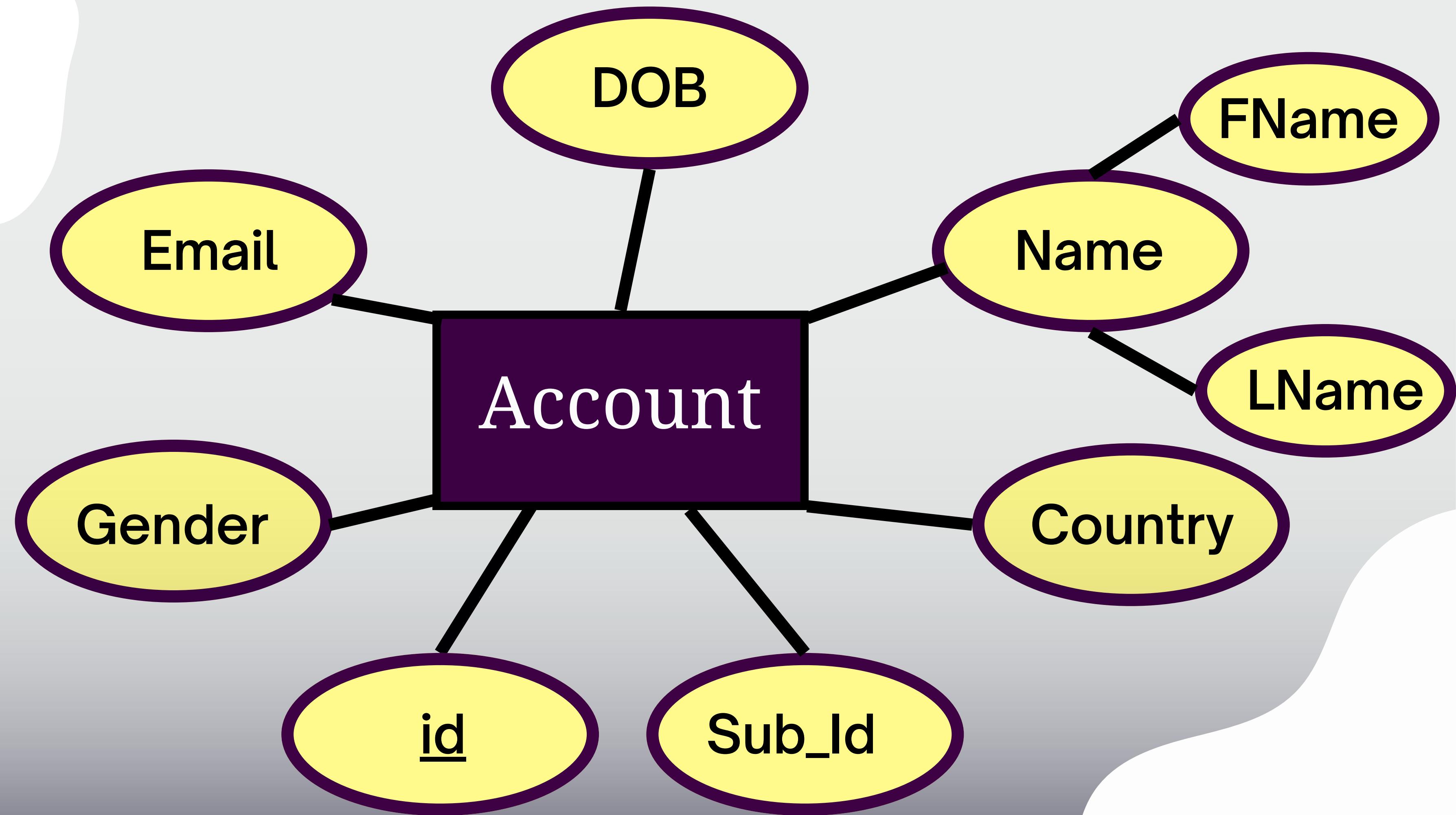
The ER model also has an associated diagrammatic representation, the ER diagram, which can express the **overall logical structure of a database graphically**.

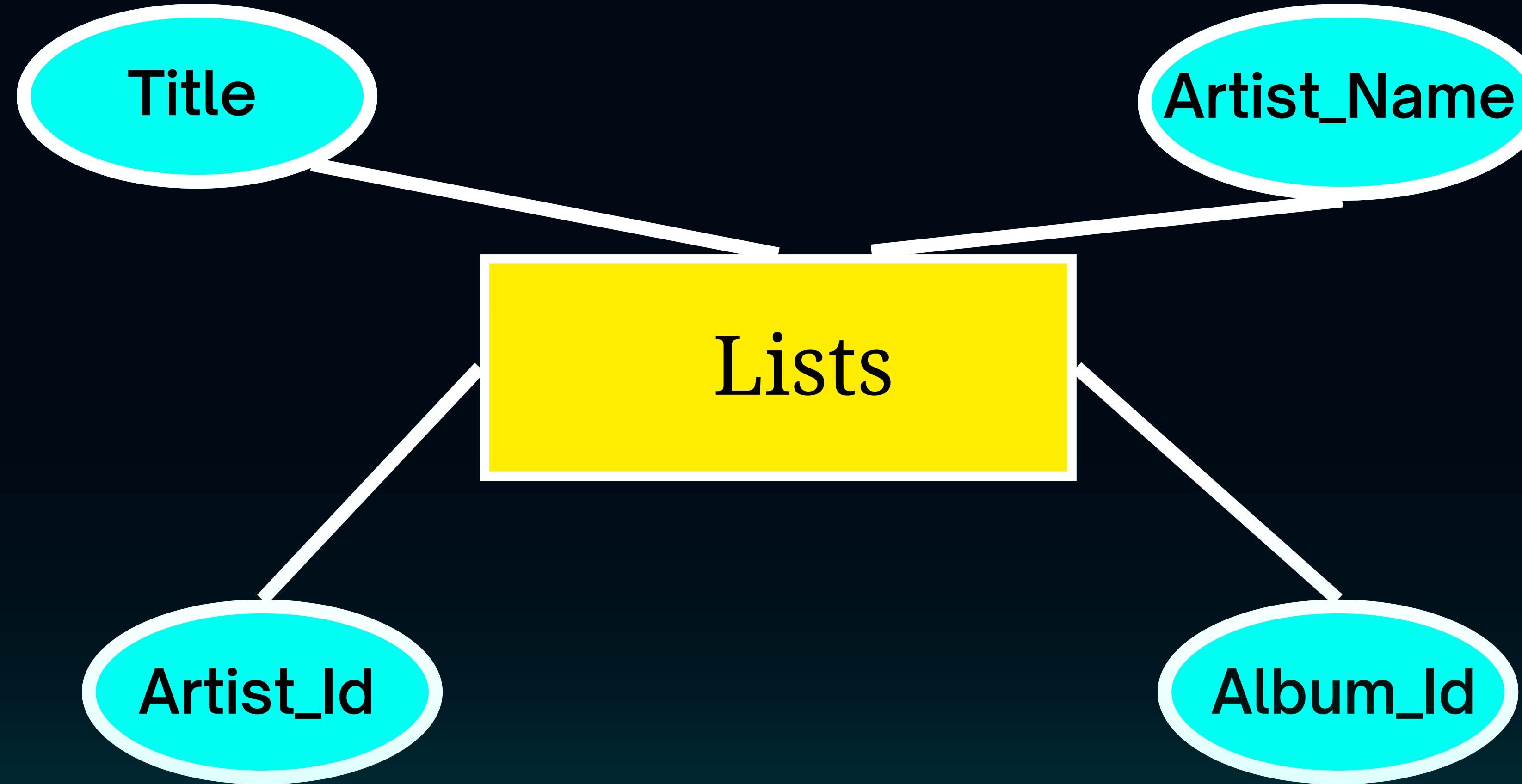




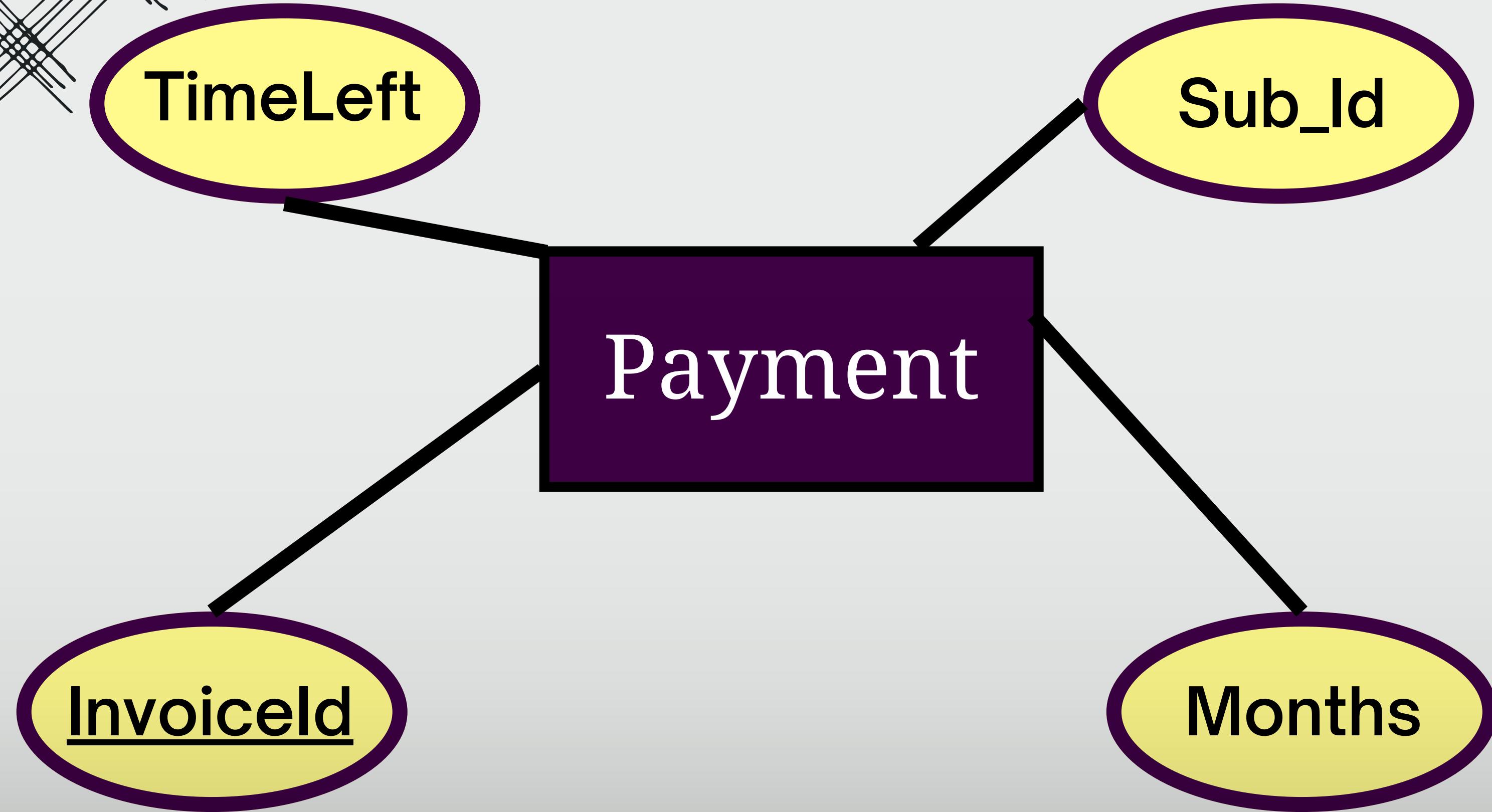
Entities And Attributes



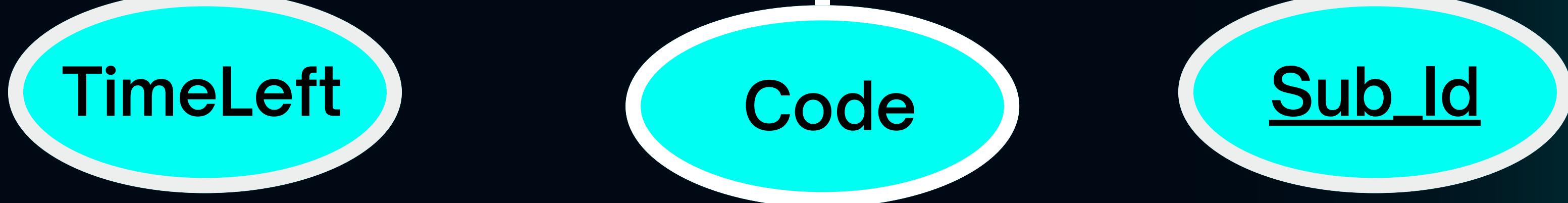


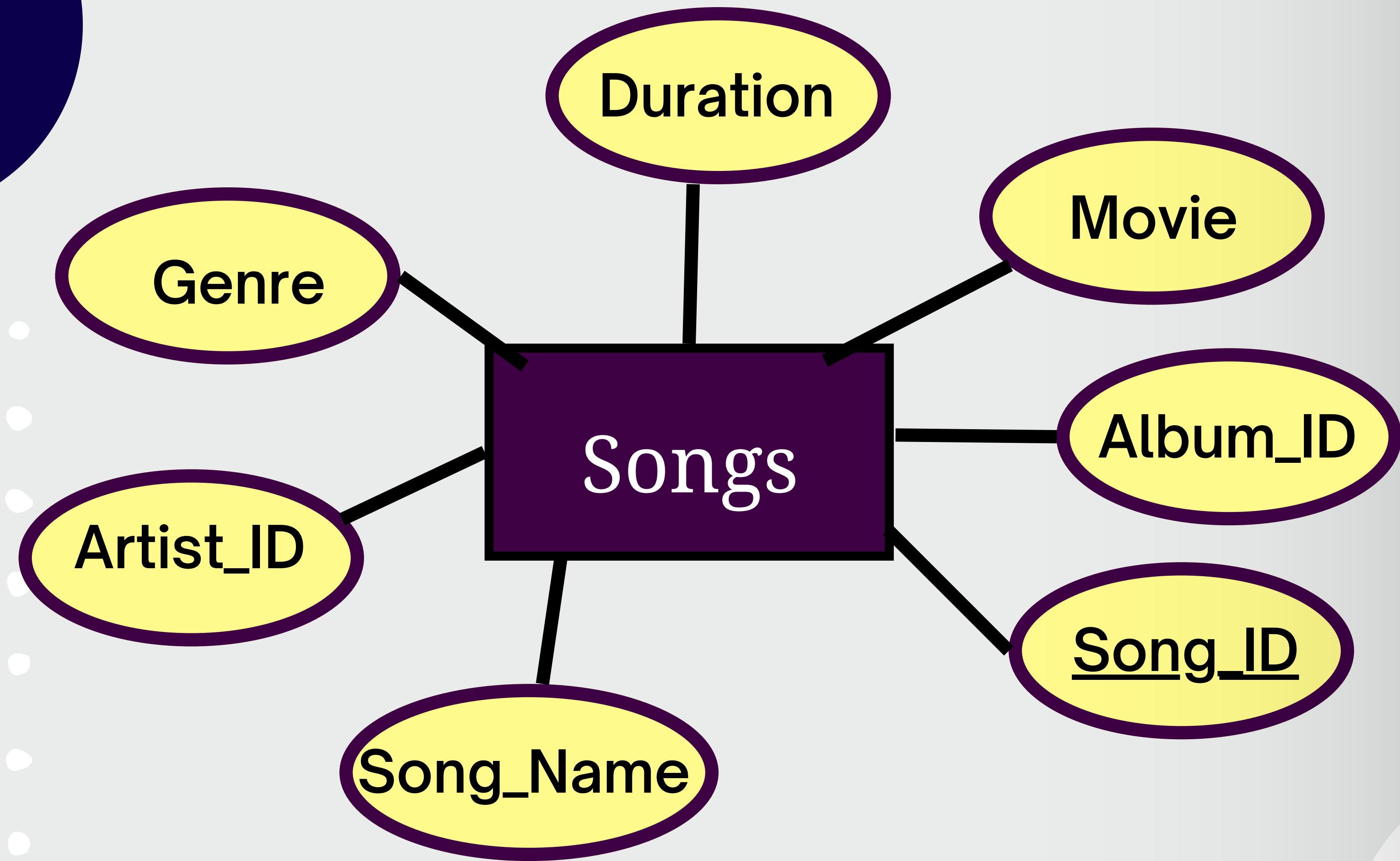


Composite key: {Artist_Id, Album_Id}



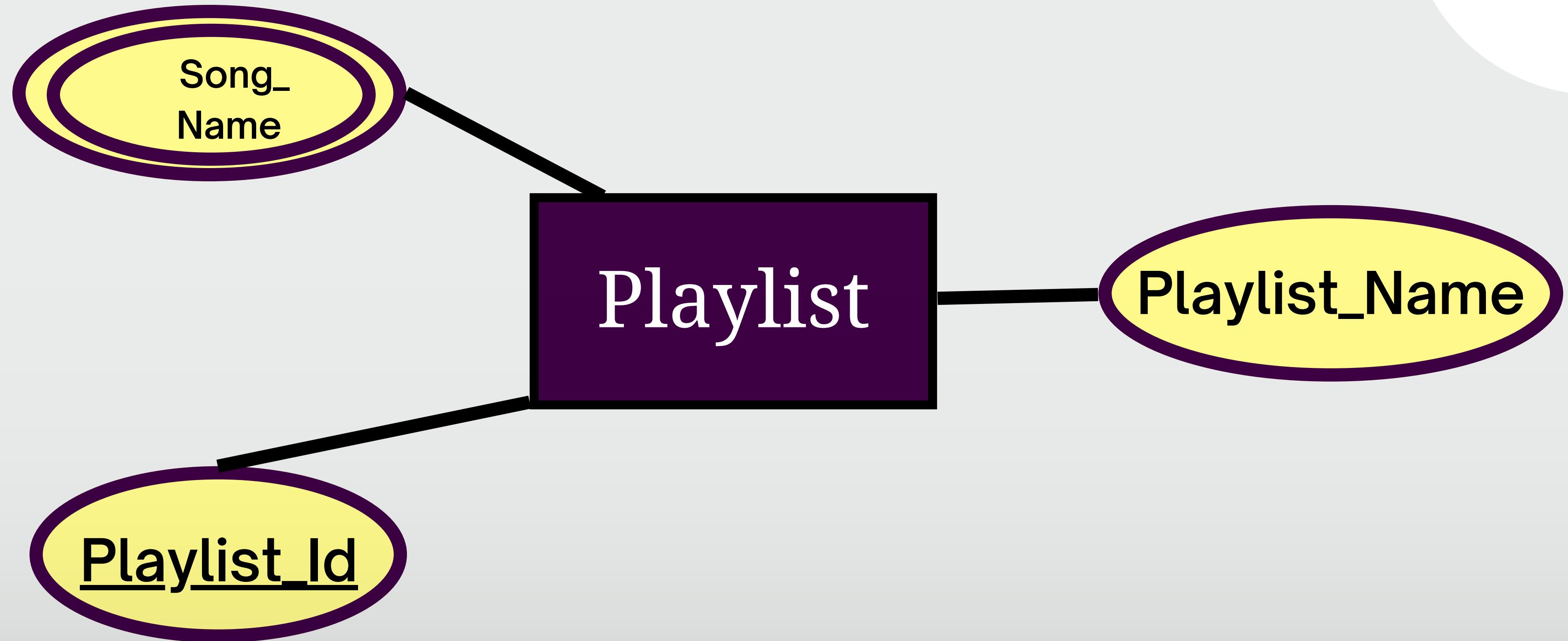
Coupons





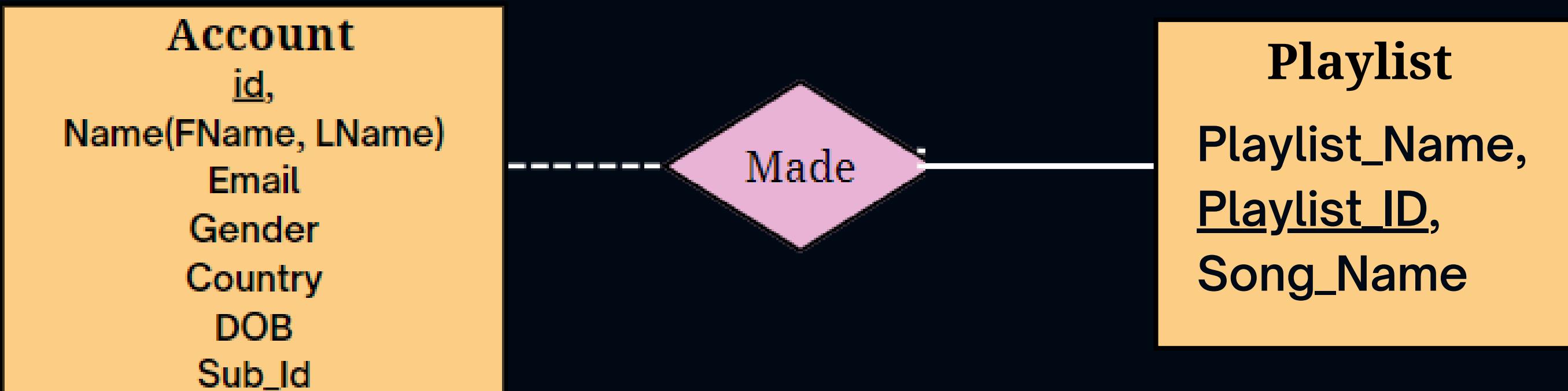
Liked





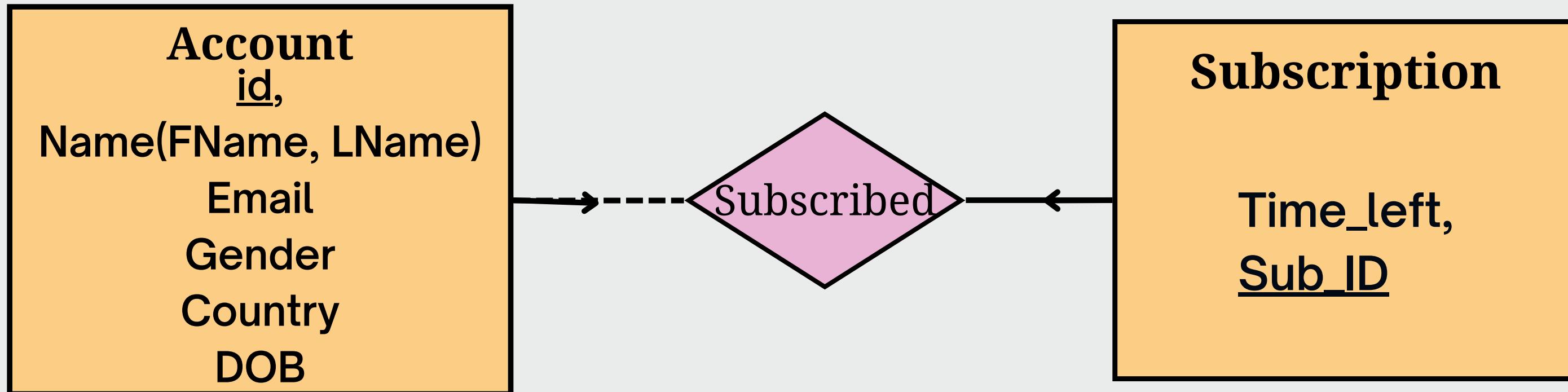
ER DIAGRAM TO TABLES





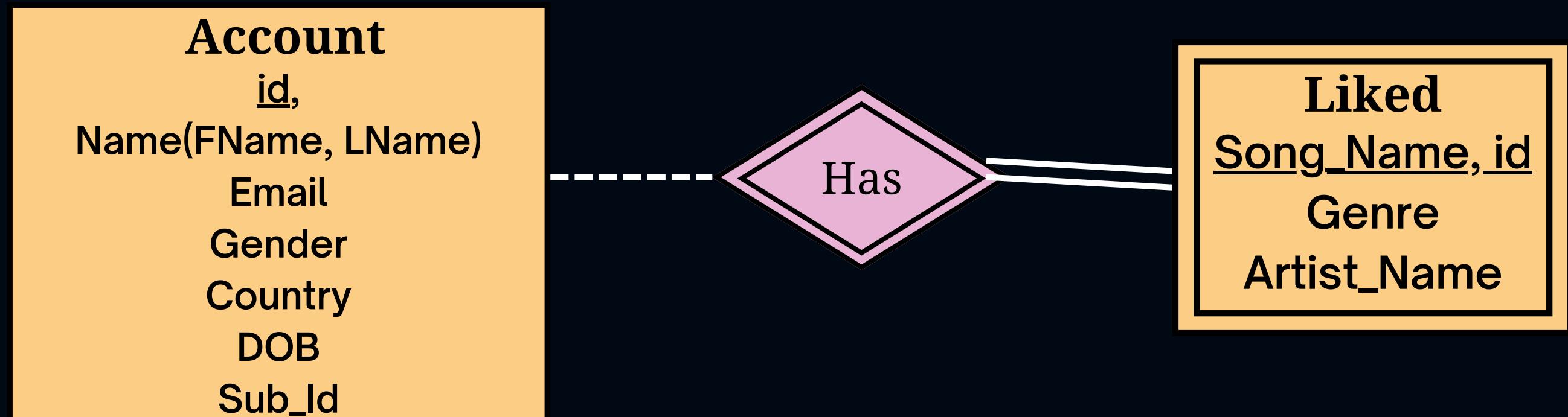
Schema: Made(Playlist_ID,id)

Composite key: Playlist ID, id



Schema: Subscribed(Sub_ID, id)

Primary Key: Can either be id or Sub_ID



Schema: Liked(**Song_Name**, **id**, Genre, Artist_Name)

Functional Dependencies: ID, Song_Name \rightarrow Song_Name, Genre, Artist_Name

Account

Id,
Name(FName, LName)
Email
Gender
Country
DOB

Schema: Account(ID, FName, LName, Email,
Gender, Country, DOB)

Functional Dependencies: ID \rightarrow Name(FName,
LName), E-mail, Gender, Country, DOB)

Album

Title,
Album_ID

Schema: Album(Title, Album_ID)

Functional Dependencies : Album_ID \rightarrow Title

Songs

Song_Name,
Genre,
Duration,
Album_ID,
Artists_ID,
Movie,
Song_ID

Schema: Songs(Song_Name, Genre, Duration,
Album_ID, Artist_ID, Movie, Song_ID)

Functional Dependencies: Song_ID → Song_Name,
Genre, Duration, Album_ID, Artist_ID

Artists

Artist_Name,
Artist_ID

Artists : (Artist_ID, Artist_Name)

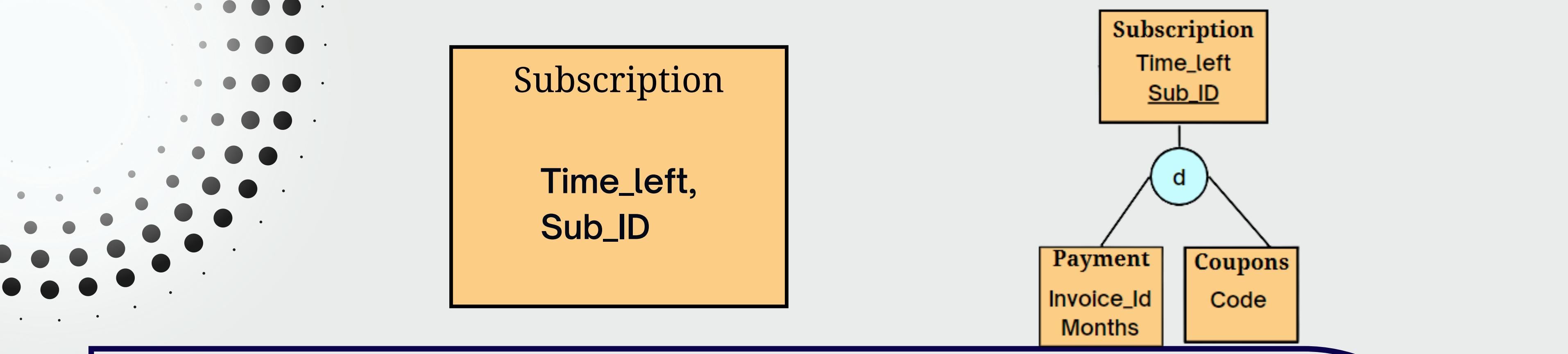
Functional Dependencies: $\text{Artist_ID} \rightarrow \text{Artist_Name}$

Playlist

Playlist_Name,
Playlist_ID,
Song_Name

Schema: Playlist(Playlist_ID, Playlist_Name, Song_Name)

Functional Dependencies: Playlist_ID \rightarrow Playlist_Name,
Song_Name



Subscription: Time_left, Sub_ID

Functional Dependencies: Sub_ID → Time_left

Schema: Payments(Invoice_ID,
Months, Time_left, Sub_ID)

Functional Dependencies :

Invoice_ID \rightarrow Months, Sub_ID, Time_left

Schema: Coupons(Code, Time_left, Sub_ID)

Functional Dependencies:

Sub_ID \rightarrow Code, Time_left

Relationship sets & Cardinality



Relationship Sets

- has :
(Song_Name, ID)
- Access:
(Song_ID, Artist_ID, Album_ID, ID)
- Made:
(ID, Playlist_ID)
- Subscribed:
(ID, sub_ID)

Cardinality

- It is a many to many relationship from liked to Account
- It is a many to many relationship from aggregate of Album and Artist to Account
- It is a many to many relationship from Playlist to Account
- It is a one to one relationship from Subscription to Account

Relationship sets and their cardinality:-

1) Has : (Song_Name, genre, Artists, Id)

‘Has’ is a Many to Many relationship from ‘Liked’ to ‘Account’ as a song can be liked from several accounts

And a single account user may like many songs as well. Here Liked shows total participation and Account

Shows partial participation as many songs can be liked by one user and there is also a possibility that he doesn't like the song as well.

2) Access_new : (Song_Id,Album_Id,Artist_Id,Id)

‘Access_new’ is a many to many relationship that connects Entity Account, Songs and Aggregation

Lists. The aggregation ‘Lists’ contains two entities Albums and Artists.

3) Made

‘Made’ is a Many to Many relationship from ‘Playlist’ to ‘Account’. As a Playlist can be used by multiple

Accounts and an account can have multiple playlists as well. Here Playlist shows total participation

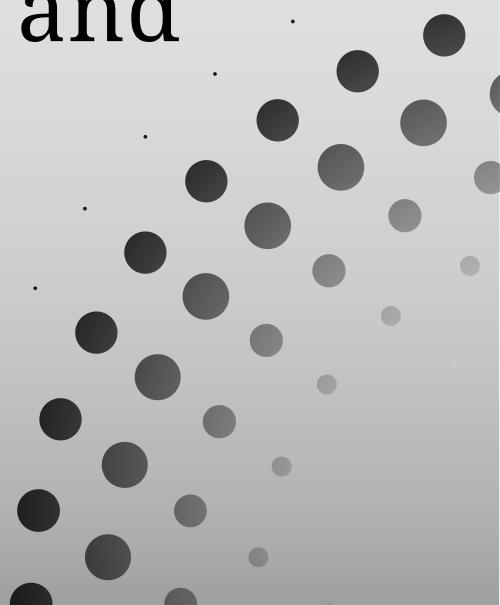
and Account shows partial participation as an account may or may not have a playlist.

4) Subscribed

‘Subscribed’ is a one to one relationship from ‘Account’ to ‘Subscription’.Here account is having

Partial participation as an account may or may not take any subscription and Subscription is having

Total participation in the relationship Subscribed.





SQL

**(Structured Query
Language)**



Account table

Table creation

```
CREATE TABLE Account (
    FName varchar(100) NOT NULL,
    LName varchar(100) NOT NULL,
    Gender varchar(10),
    Email varchar(50) NOT NULL,
    Country varchar(15) NOT NULL,
    DOB date NOT NULL,
    id int PRIMARY KEY,
);
```

Output

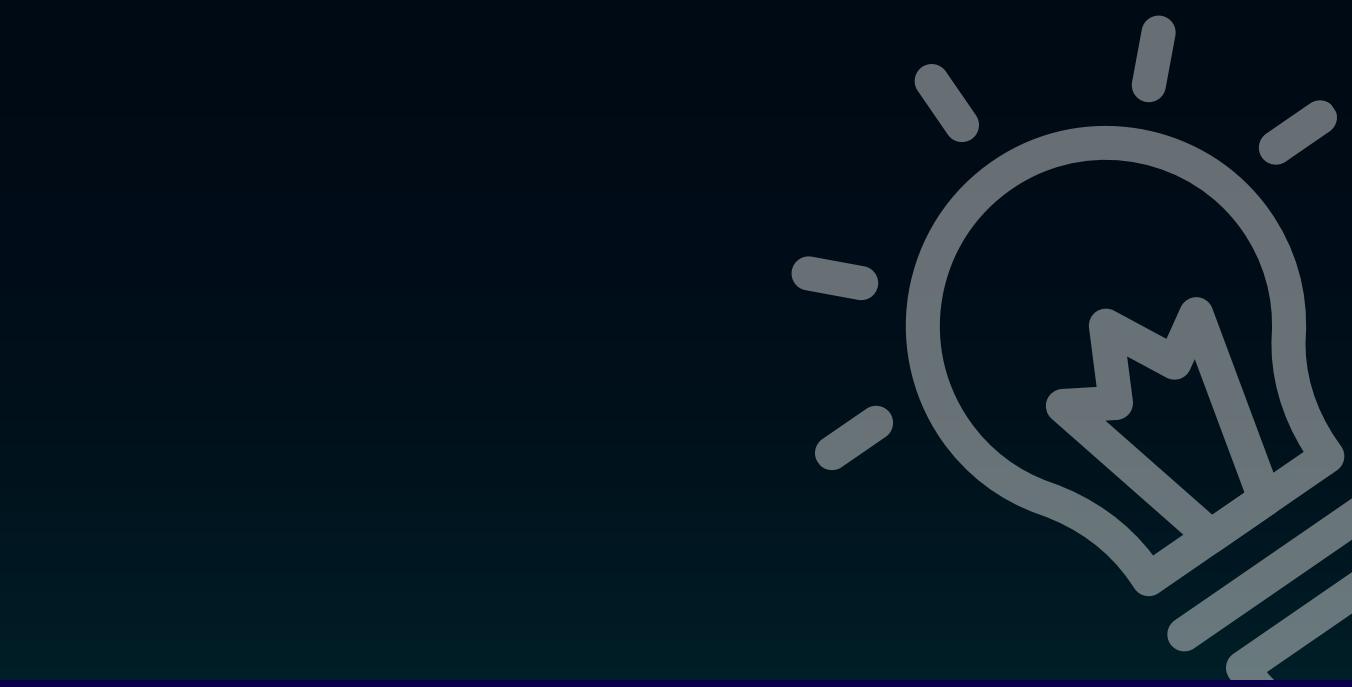
	Field	Type	Null	Key	Default
▶	FName	varchar(100)	NO		NULL
	LName	varchar(100)	YES		NULL
	Country	varchar(25)	NO		NULL
	id	char(100)	NO	PRI	NULL
	DOB	date	NO		NULL
	Email	varchar(50)	NO		NULL
	Gender	varchar(10)	YES		NULL

Insertion

```
INSERT INTO Account VALUES('Harry', 'Potter', 'Male', 'harry@gmail.com', 'Britain', '2002-02-25', 100, 30);
INSERT INTO Account VALUES('Ram', 'Singh', 'Male', 'ram@gmail.com', 'India', '2002-01-31', 101, 999);
INSERT INTO Account VALUES('Shyam', 'Rathore', 'Male', 'Shyaam@gmail.com', 'India', '1999-11-30', 102, NULL);
INSERT INTO Account VALUES('Ana', 'Armas', 'Female', 'Ana@gmail.com', 'USA', '2004-03-18', 103, NULL);
INSERT INTO Account VALUES('Aditi', 'Mishra', 'Female', 'Aditi@gmail.com', 'India', '2001-02-16', 104, 263);
INSERT INTO Account VALUES('Rajiv', 'Singh', 'Male', 'rajiv@gmail.com', 'India', '2000-05-25', 105, 256);
INSERT INTO Account VALUES('Ishita', 'Shukla', 'Female', 'ishita@gmail.com', 'India', '2003-11-30', 106, NULL);
INSERT INTO Account VALUES('Alexia', 'Putellas', 'Female', 'alexia@gmail.com', 'Spain', '2002-05-13', 107, 2);
INSERT INTO Account VALUES('Luke', 'Jing', 'Male', 'luke@gmail.com', 'New Zealand', '1990-09-21', 108, 252);
INSERT INTO Account VALUES('Elssye', 'Perry', 'Female', 'elperry@gmail.com', 'Australia', '1992-12-25', 109, 955);
```

OUTPUT

	FName	LName	Gender	Email	Country	DOB	id	Sub_Id
▶	Harry	Potter	Male	harry@gmail.com	Britain	2002-02-25	100	30
	Ram	Singh	Male	ram@gmail.com	India	2002-01-31	101	999
	Shyam	Rathore	Male	Shyaam@gmail.com	India	1999-11-30	102	NULL
	Ana	Armas	Female	Ana@gmail.com	USA	2004-03-18	103	NULL
	Aditi	Mishra	Female	Aditi@gmail.com	India	2001-02-16	104	263
	Rajiv	Singh	Male	rajiv@gmail.com	India	2000-05-25	105	256
	Ishita	Shukla	Female	ishita@gmail.com	India	2003-11-30	106	NULL
	Alexia	Putellas	Female	alexia@gmail.com	Spain	2002-05-13	107	2
	Luke	Jing	Male	luke@gmail.com	New Zealand	1990-09-21	108	252
	Elssye	Perry	Female	elperry@gmail.com	Australia	1992-12-25	109	955
	NULl	NULl	NULl	NULl	NULl	NULl	NULl	NULl



Account table

songs

Table creation

```
create table songs (
Song_Id int PRIMARY key,
Genre varchar(25) ,
Song_Name varchar(25) NOT NULL,
Movie varchar(25),
Duration varchar(15) NOT NULL
);
```

Output

	Field	Type	Null	Key	Default
▶	Song_Id	int	NO	PRI	HULL
	Genre	varchar(25)	YES		HULL
	Song_Name	varchar(25)	NO		HULL
	Movie	varchar(25)	YES		HULL
	Duration	varchar(15)	NO		HULL

Insertion

```
INSERT INTO songs VALUES(200, 'Rock', 'In  
Dino','Interstellar', '00:02:59');  
  
INSERT INTO songs VALUES(201, 'Rock', 'Bailando', Null,  
'00:02:54');  
  
INSERT INTO songs VALUES(202, 'Pop', 'Skyfall','Skyfall',  
'00:03:54');  
  
INSERT INTO songs VALUES(203, 'Jazz', 'No Lie', 'No Time',  
'00:03:54');  
  
INSERT INTO songs VALUES(204, 'Classical', 'Ilahi', 'Roy',  
'00:02:54');  
  
INSERT INTO songs VALUES(205, 'Blues', 'Tu Hi', 'Roy',  
'00:03:22');  
  
INSERT INTO songs VALUES(206, 'Classical', 'Duniyaa', Null,  
'00:04:22');  
  
INSERT INTO songs VALUES(207, 'Blues', 'Tum Ho', Null,  
'00:04:29');  
  
INSERT INTO songs VALUES(208, 'Ghazal', 'Garaj Baras',  
'Insight', '00:01:59');  
  
INSERT INTO songs VALUES(209, 'Ghazal', 'Badi Nazuk',  
'Insight', '00:03:29');  
  
select * from songs;
```

OUTPUT

	Song_Id	Genre	Song_Name	Movie	Duration
▶	200	Rock	In Dino	Interstellar	00:02:59
	201	Rock	Bailando	NULL	00:02:54
	202	Pop	Skyfall	Skyfall	00:03:54
	203	Jazz	No Lie	No Time	00:03:54
	204	Classical	Ilahi	Roy	00:02:54
	205	Blues	Tu Hi	Roy	00:03:22
	206	Classical	Duniyaa	NULL	00:04:22
	207	Blues	Tum Ho	NULL	00:04:29
	208	Ghazal	Garaj Baras	Insight	00:01:59
	209	Ghazal	Badi Nazuk	Insight	00:03:29
●	HULL	HULL	HULL	HULL	HULL

Payments

Table creation

```
create table Payments(  
    Time_Left int NOT NULL,  
    Invoice_Id varchar(100),  
    Sub_Id varchar(100) NOT NULL,  
    Months int NOT NULL);
```

Outputs

	Field	Type	Null	Key	Default	Extra
▶	Time_Left	int	NO		NULL	
	Invoice_Id	varchar(100)	NO	PRI	NULL	
	Sub_Id	varchar(100)	NO		NULL	
	Months	int	NO		NULL	

Insertion

```
insert into Payments values( 4, '123', '091', 5);
insert into Payments values( 5, '780', '782', 10);
insert into Payments values( 1, '230', '112', 7);
insert into Payments values( 4, '333', '111', 9 );
insert into Payments values( 3, '112', '234', 12);
insert into Payments values( 2 , '300', '235', 12 );
insert into Payments values( 5, '145', '671', 10 );
insert into Payments values( 1, '545', '187', 2 );
insert into Payments values( 3, '672', '441', 10 );
insert into Payments values( 3, '705', '342', 4 );
select * from Payments;
```

Output

	Time_Left	Invoice_Id	Sub_Id	Months
▶	125	1	0	10
	3	112	234	12
	4	123	091	5
	5	145	671	10
	1	230	112	7
	2	300	235	12
	4	333	111	9
	1	545	187	2
	697	6	3	12
	3	672	441	10
	3	705	342	4
	5	780	782	10
	347	9	4	5

Access_new

Table creation

```
create table Access_new  
( Song_Id numeric(3,0)  
NOT NULL,  
    Artist_Id numeric(2,0)  
NOT NULL,  
    Album_Id numeric(2,0)  
NOT NULL );
```

Output

Field	Type	Null	Key
Song_Id	decimal(3,0)	NO	
Artist_Id	decimal(2,0)	NO	
Album_Id	decimal(2,0)	NO	

Insertion

```
insert into Access_new values( 200 , 10 , 50 );
insert into Access_new values( 201 , 11 , 51 );
insert into Access_new values( 202 , 12 , 52 );
insert into Access_new values( 203 , 12 , 53 );
insert into Access_new values( 204 , 13 , 54 );
insert into Access_new values( 205 , 13 , 54 );
insert into Access_new values( 206 , 15 , 55 );
insert into Access_new values( 207 , 13 , 55 );
insert into Access_new values( 208 , 18 , 56 );
insert into Access_new values( 209 , 18 , 56 );
```

Output

Song_Id	Artist_Id	Album_Id
200	10	50
200	10	50
201	11	51
202	12	52
203	12	53
204	13	54
205	13	54
206	15	55
207	13	55
208	18	56
209	18	56

Playlists

Table creation

```
create table playlists
( playlist_Id numeric(3,0) NOT NULL,
  song varchar(100) NOT NULL,
  playlist_name varchar(100) NOT NULL);
```

Output

Field	Type	Null	Key	Default
playlist_Id	decimal(3,0)	NO		NULL
song	varchar(100)	NO		NULL
playlist_name	varchar(100)	NO		NULL

Insertion

```
insert into playlists values( 400 , "Bailando" , "P10");
insert into playlists values( 400 , "In Dino" , "P10");
insert into playlists values( 401 , "In Dino" , "P11");
insert into playlists values( 402 , "Badi Nazuk" , "P13");
insert into playlists values( 402 , "Garaj Baras" , "P13");
insert into playlists values( 403 , "Bailando" , "P15");
insert into playlists values( 403 , "Skyfall" , "P15");
insert into playlists values( 404 , "Bailando" , "P14");
insert into playlists values( 405 , "Tum ho" , "P16");
insert into playlists values( 406 , "Duniya" , "P17");
insert into playlists values( 407 , "Duniya" , "P18");
insert into playlists values( 407 , "Ilahi" , "P18");
insert into playlists values( 407 , "No lie" , "P18");
```

Output

playlist_Id	song	playlist_name
400	Bailando	P10
400	In Dino	P10
401	In Dino	P11
402	Badi Nazuk	P13
402	Garaj Baras	P13
403	Bailando	P15
403	Skyfall	P15
404	Bailando	P14
405	Tum ho	P16
406	Duniya	P17
407	Duniya	P18
407	Ilahi	P18
407	No lie	P18

Album

Table creation

```
create table Album  
( Album_Id numeric(2,0)  
    PRIMARY KEY,  
Title Varchar(100) NOT  
    NULL );
```

Output

Field	Type	Null	Key	Default
Album_Id	decimal(2,0)	NO	PRI	NULL
Title	varchar(100)	NO		NULL

Insertion

```
insert into Album values( 50, "Hello");
insert into Album values( 51, "Hind");
insert into Album values( 52, "Bird");
insert into Album values( 53, "Crow");
insert into Album values( 54, "Fifty");
insert into Album values( 55, "Visca barka");
insert into Album values( 56, "Viral");
```

Output

Album_Id	Title
50	Hello
51	Hind
52	Bird
53	Crow
54	Fifty
55	Visca barka
56	Viral

Artists

Table creation

```
create table Artists (
Artist_ID varchar(15) Primary key,
Artist_name varchar(25) NOT NULL
);
```

Output

	Field	Type	Null	Key	Default
▶	Artist_ID	varchar(15)	NO	PRI	NUL
	Artist_name	varchar(25)	NO		NUL

Insertion

```
INSERT INTO Artists VALUES('46a1', 'Adele');
INSERT INTO Artists VALUES('45d6', 'Enrique Iglesias');
INSERT INTO Artists VALUES('48b7', 'Sonu Nigam');
INSERT INTO Artists VALUES('4aab', 'KK');
INSERT INTO Artists VALUES('4aab', 'Asha Bhosle');
INSERT INTO Artists VALUES('4aab', 'Jagjit Singh');
```

```
select * from Artist;
```

Output

	Artist_ID	Artist_name
▶	45d6	Enrique Iglesias
	46a1	Adele
	48b7	Sonu Nigam
	4aab	KK
	4da1	Adele
	NULL	NULL

Coupon

Table creation code

```
create table Coupons (
    Time_Left int NOT NULL,
    Sub_Id int PRIMARY KEY,
    Codee varchar(20) NOT NULL
);
insert into Coupons values (6,30,'AED23');
insert into Coupons values (7,252,'LLL23');
insert into Coupons values (3,256,'AED23');
insert into Coupons values (3,263,'KK233');
select * from coupon;
```

Output

	Field	Type	Null	Key	Default
▶	Time_Left	int	NO		HULL
▶	Sub_Id	int	NO	PRI	HULL
▶	Codee	varchar(20)	NO		HULL

	Time_Left	Sub_Id	Codee
▶	6	30	AED23
▶	7	252	LLL23
▶	3	256	AED23
▶	3	263	KK233
	HULL	HULL	HULL

liked

```
create table liked  
(  
    song_name varchar(30),  
    artist_name varchar (20),  
    genre  varchar (10)  
);
```

	Field	Type	Null	Key	Default	Extra
▶	song_name	varchar(30)	YES		NULL	
	artist_name	varchar(20)	YES		NULL	
	genre	varchar(10)	YES		NULL	

Insertion

```
insert into liked values ( 'In dino ', ' Adele ', 'Rock ' );
insert into liked values ( 'In dino ', ' Adele ', 'Rock ' );
insert into liked values ( 'Tu hi ', ' KK ', 'Blues ' );
insert into liked values ( 'Badi Najuk ', ' Jagjit Singh ', 'Ghajal ' );
insert into liked values ( 'Tu hi ', ' KK ', 'Blues ' );
insert into liked values ( 'In dino ', ' Adele ', 'Rock ' );
insert into liked values ( 'Tu hi ', ' KK ', 'Blues ' );
insert into liked values ( 'Ilahi ', ' KK ', 'Blues ' );
insert into liked values ( 'Ilahi ', ' KK ', 'Blues ' );
insert into liked values ( 'Tum ho ', ' Adele ', 'Rock ' );
insert into liked values ( 'Garaj baras ', ' Jagjit singh ', 'Ghazal ' );
insert into liked values ( 'Badi Nazuk ', ' Jagjit singh ', 'Ghazal ' );
insert into liked values ( 'Duniya ', ' Asha Bhosle ', 'classical ' );
insert into liked values ( 'Sky Fall ', ' Sonu Nigam ', 'Pop ' );
insert into liked values ( 'Bailnaldo ', ' Enrique Iglesias ', 'Rock ' );
insert into liked values ( 'Duniya ', ' Asha Bhosle ', 'classical ' );
select * from liked ;
```

Output

	song_name	artist_name	genre
	Duniya	Asha Bhosle	classical
	In dino	Adele	Rock
	In dino	Adele	Rock
	Tu hi	KK	Blues
	Badi Najuk	Jagjit Singh	Ghajal
	Tu hi	KK	Blues
	In dino	Adele	Rock
	Tu hi	KK	Blues
	Ilahi	KK	Blues
	Ilahi	KK	Blues
	Tum ho	Adele	Rock
	Garaj baras	Jagjit singh	Ghazal
	Badi Nazuk	Jagjit singh	Ghazal
	Duniya	Asha Bhosle	classical
	Sky Fall	Sonu Nigam	Pop
	Bailnaldo	Enrique Iglesias	Rock
	Duniya	Asha Bhosle	classical

Made

Made

```
create table Made  
( Playlist_Id numeric(3,0)  
NOT NULL,  
Id numeric(3,0) );
```

Output

Field	Type	Null	Key
Playlist_Id	decimal(3,0)	NO	
Id	decimal(3,0)	YES	

Insertion

```
insert into Made values( 400 , 100);
insert into Made values( 401 , 100);
insert into Made values( 402 , 101);
insert into Made values( 403 , 102);
insert into Made values( 404 , 103);
insert into Made values( 405 , 103);
insert into Made values( 406 , 106);
insert into Made values( 407 , 109);
```

Output

Playlist_Id	Id
400	100
401	100
402	101
403	102
404	103
405	103
406	106
407	109



Normalisation



Lists

Lists (Artist_Id, Artist_Name, Album_Id, Title)

Primary Key: {Artist_Id, Album_Id}

FD: Album_Id → Title

Artist_Id → Artist_Name

Album_Id, Artist_Id → Title, Artist_Name

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: Title is depending on Album_Id and Artist_Name depending on Artist_Id. Hence, there is partial dependency as part of the primary key determines non-prime attributes.

So, on decomposing we get,

Artists (Artist_Id, Artist_name)

Album (Album_Id, title)

Step 3: In the first entity there are no non-prime attributes so transitive dependency is not possible.

And in second primary key determines both the non-prime attributes so, transitive Dependency is not there. Therefore, both the entities are in 3NF.

Step 4: As in both the entities Primary keys are on the left hand side. Therefore, The schema is in BCNF form.

Liked

Liked (Genre, Song_Name, Artists, Id)

Primary Key: {Id, Song_Name}

Discriminator: Song_Name

FD: Id, Song_Name → Genre, Artists

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: As all the attributes of the weak entity are in the primary key, there is no partial dependency.

So, the schema is 2NF.

Step 3: There are no non-prime attributes so transitive dependencies do not exist. So, the schema

Is in 3NF.

Step 4: As in all FD's the left-hand side is the super key. So the schema is also in BCNF form

Coupon

Coupon (Time_Left, Sub_Id, Code)

Primary Key: Sub_Id

FD:

$\text{Sub_Id} \rightarrow \text{Code, Time_Left}$

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

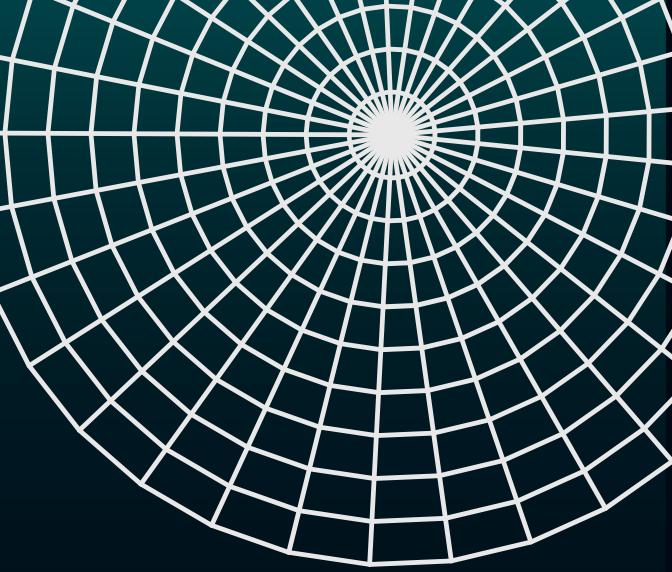
Step 2: As Sub_Id is a simple key. So, the schema is also in 2NF.

$\text{Sub_Id} \rightarrow \text{all attributes}$

Step 3: As no transitive dependencies between non-prime attributes exist. So, the schema is in 3NF too.

Step 4: As in all FD's the left-hand side is the super key. So the schema is also in BCNF form

Payments



Payments (Time_Left, Invoice_Id, Sub_Id, Months)

Primary Key : Invoice_Id

FD : Invoice_Id → Months, Time_Left, Sub_Id
Sub_Id → Months, Time_Left

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: As Id is a simple key. So, the schema is also in 2NF.

Step 3: As no transitive dependencies between non-prime attributes exist. So, the schema is in 3NF too.

Step 4: As in all FD's the left-hand side is the super key. So the schema is also in BCNF form

Songs

Songs (Duration, Song_Id, Movie, Genre, Song_Name)

Primary Key: Song_Id

FD : Song_Id → Duration, Movie, Genre, Song_Name

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: As Id is a simple key. So, the schema is also in 2NF.

Song_Id → all attributes

Step 3: As no transitive dependencies between non-prime attributes exist. So, the schema is in 3NF too.

Step 4: As in all FD's the left-hand side is the super key. So now both the schemas are in BCNF form.

Playlist

Playlist (Song, Playlist_Id, Playlist_Name)

Primary key: {Playlist_Id, Song}

FD: Playlist_Id, Song → Playlist_Name

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: As playlist_Id is a simple key. So, the schema is also in 2NF.

playlist_Id, Song → all attributes

Step 3: As no transitive dependencies between non-prime attributes exist. So, the schema is in 3NF too.

Step 4: As in above FD the left-hand side is the super key. So the schema is also in BCNF form

Account

Account (Id, DOB, Gender, Email, L_Name, F_Name, Country, Sub_Id)

Primary Key : Id

Foreign Key: Sub_Id

FD: Id → DOB, Gender, Email, L_Name, F_Name, Country, Sub_Id

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: As Id is a simple key. So, the schema is also in 2NF.

Id → all attributes

Step 3: As no transitive dependencies between non-prime attributes exist. So, the schema is in 3NF too.

Step 4: As in all FD's the left-hand side is the super key. So the schema is also in BCNF form

Made

Made (Id, Playlist Id)

Primary Key : Playlist_Id

FD: Playlist_Id → Id

Step 1: Assuming both the attributes have an atomic domain, the schema is in 1NF.

Step 2: As Playlist_Id is a simple key. So, the schema is also in 2NF.

Playlist_Id → Id

Step 3: As there is only one non-prime attribute so no transitive dependency can exist. Therefore,

The schema is in 3NF form.

Step 4: As in the given FD the left-hand side is the Primary key. So the schema is also in BCNF form.

Access_new



Access_new(Song_Id, Artist_Id, Album_Id, Id)

Primary Key : {Song_Id, Id}

FD : Song_Id → Artist_Id, Album_Id
Song_Id, Id → Artist_Id, Album_Id

Step 1: Assuming all the attributes have an atomic domain, the schema is in 1NF.

Step 2: As part of the primary key determines non-prime attributes. So, there is partial dependency
And therefore it is not in 2NF.

Now on decomposing entity we get:

Access_new (Song_Id, Id),

Access_new1(Song_Id, Artist_Id, Album_Id)

Now, both the entities are in 2NF.

Step 3: In the first entity there are no non-prime attributes so transitive dependency is not possible.

And in second primary key determines both the non-prime attributes so, transitive
Dependency is not there. Therefore, both the entities are in 3NF.

Step 4: As in both the entities Primary keys are on the left hand side. Therefore, The schema
Is in BCNF form.



SQL Queries



Query 1

- Find the names of the most liked artist.

```
mysql> select artists, max(total) as liked_by
-> from ( select artists, count(artists) as total
-> from liked
-> group by artists
-> order by count(artists) desc
-> ) as a;
```

artists	liked_by
KK	6

Query 2

- Find all the songs in the playlist made by “Elssye Perry”.

```
mysql> select playlist.song
-> from playlist,made, account
-> where account.Fname="Elssye" and account.Lname="Perry" and account.id=made.id and made.playlist_id=playlist.playlist_id;
+-----+
| song |
+-----+
| Duniyaa |
| Ilahi |
| No Lie |
+-----+
```

Query 3

- Name of people who use “AED23” as code for subscription.

```
mysql> select account.Fname, account.Lname
-> from coupon, account
-> where coupon.code="AED23" and coupon.Sub_Id=account.Sub_Id;
+-----+-----+
| Fname | Lname |
+-----+-----+
| Harry | Potter |
| Rajiv | Singh |
+-----+-----+
```

Query 4

- Name other songs of the album “Viral” other than song “Garaj Baras”.

```
mysql> select Song_Name
    -> from( select Song_name
    -> from access_new, songs, album
    -> where Title="Viral" and album.Album_Id=Access_new.Album_Id
    -> and access_new.Song_Id=songs.Song_Id) as a
    -> where Song_Name != "Garaj Baras";
+-----+
| Song_name |
+-----+
| Badi Nazuk |
+-----+
```

Query 5

- Name all the songs and their description, which starts with “B” and ends with “O”.

```
mysql> select *
    -> from Songs
    -> where Song_Name Like "B%O";
+-----+-----+-----+-----+
| Song_Id | Genre | Song_Name | Movie | Duration |
+-----+-----+-----+-----+
| 201 | Rock | Bailando | NULL | 00:02:54 |
+-----+-----+-----+-----+
```

Relational Algebra

Query 1:

- *Find the names and sub_id of the customers belonging to India.*

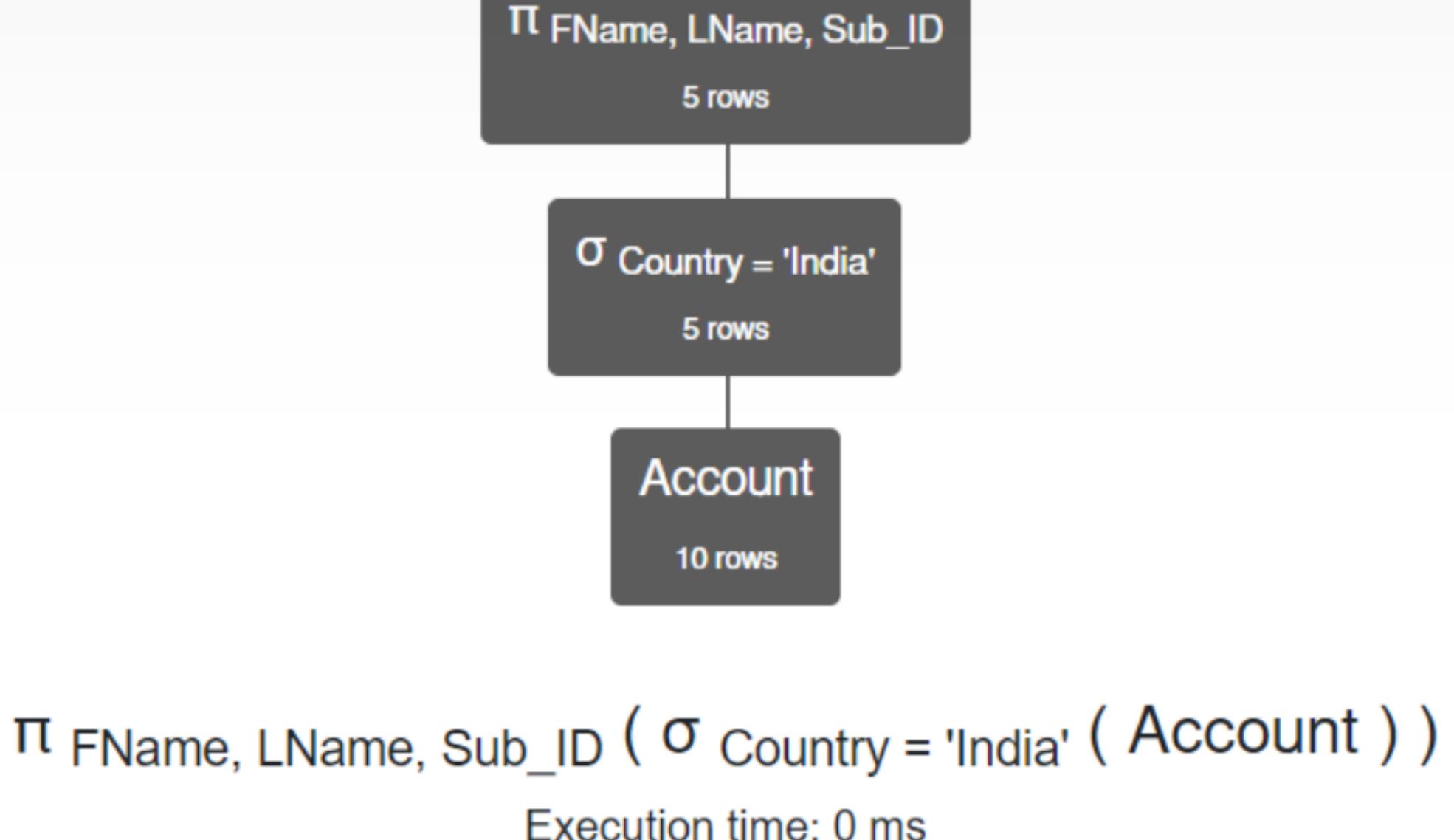
Input

Account (Id, DOB,
Gender, E-mail,
L_name,F_name,Coun
try,Sub_Id)

Output:

$\pi \text{ FName, LName, Sub_Id } (\sigma$
 $\text{Country} = \text{'India'} (\text{Account}))$

Output



Account.FName	Account.LName	Account.Sub_ID
'Ram'	'Singh'	'999'
'Shyam'	'Singh'	'NULL'
'Aditi'	'Mishra'	'263'
'Rajiv'	'Singh'	'256'
'Ishita'	'Shukla'	'NULL'

Query 2:

- Find the E-mail IDs of the customers whose subscribed time is less than 5 days.

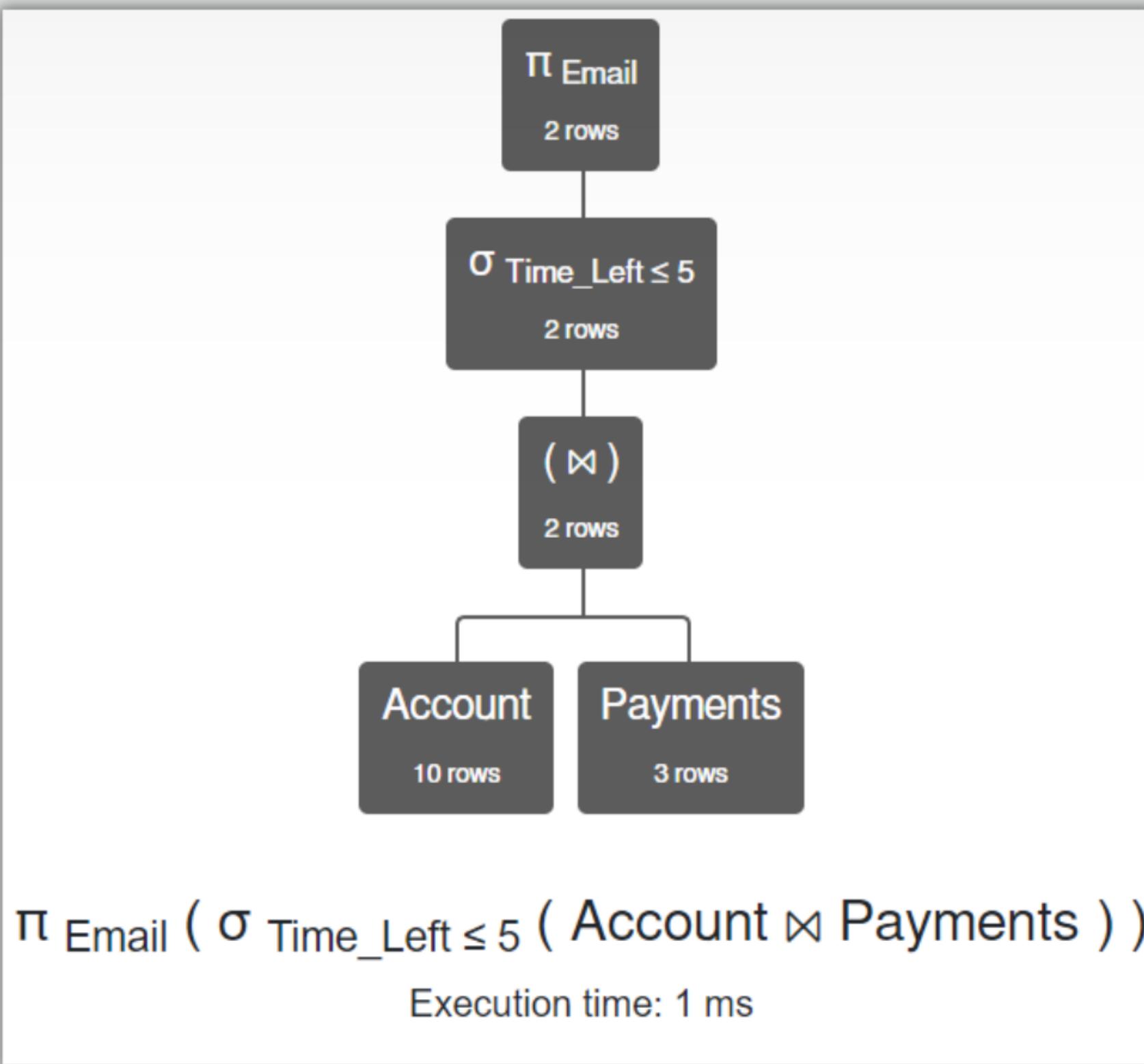
Input

Account (Id, DOB, Gender, E-mail,L-name,F-name,Country,Sub-Id)
Payment(Time_Left, Invoice_ID,Sub_Id,,Months)

Output

$\pi \text{ Email} (\sigma \text{ Time_Left} \leq 5 \text{ (Account Payments)})$

Output



Account.Email

'ram@gmail.com'

'alexi@gmail.com'

Query 3:

- List the names of the songs and their artists and the DOB of the female users who have 'liked' the songs.

Input

Account(Id, DOB,

Gender,E-

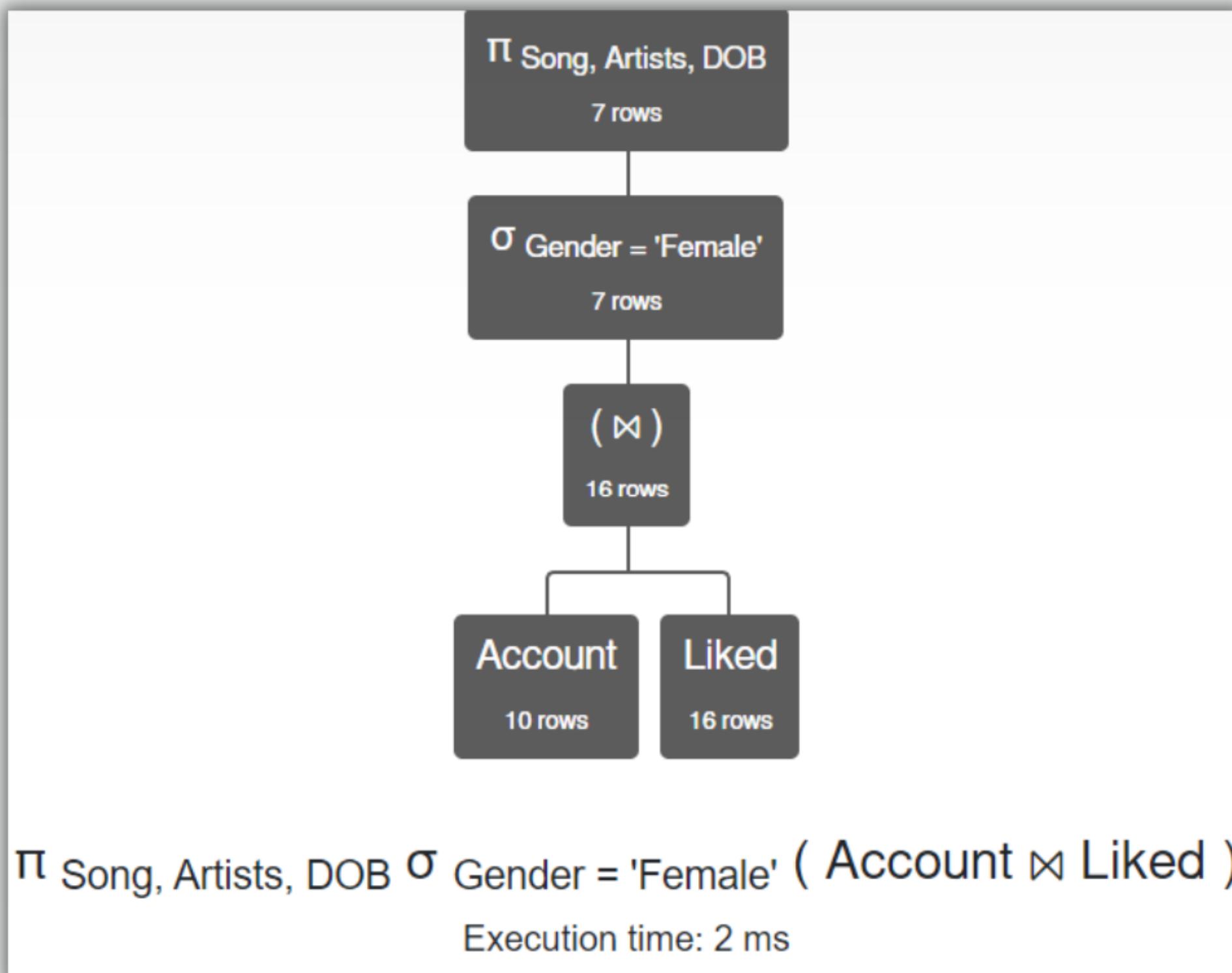
mail,L_name,F_name,Coun

try,Sub_Id)

Output

π Song, Artists, DOB σ
Gender = 'Female'
(Account Liked)

Output



Liked.Song	Liked.Artists	Account.DOB
'In Dino'	'Adele'	'2002-01-30'
'Tu Hi'	'KK'	'2002-01-30'
'Ilahi'	'KK'	'2003-02-06'
'Garaj Baras'	'Jagjit Singh'	'2000-02-06'
'Badi Nazuk'	'Jagjit Singh'	'2002-01-14'
'Duniyaa'	'Asha Bhosle'	'2002-01-14'
'Skyfall'	'Sonu Nigam'	'2002-01-14'

Query 4:

- Find the Song_Id, Album_Id, and Song_Name of Songs have been sung by Jagjit Singh .

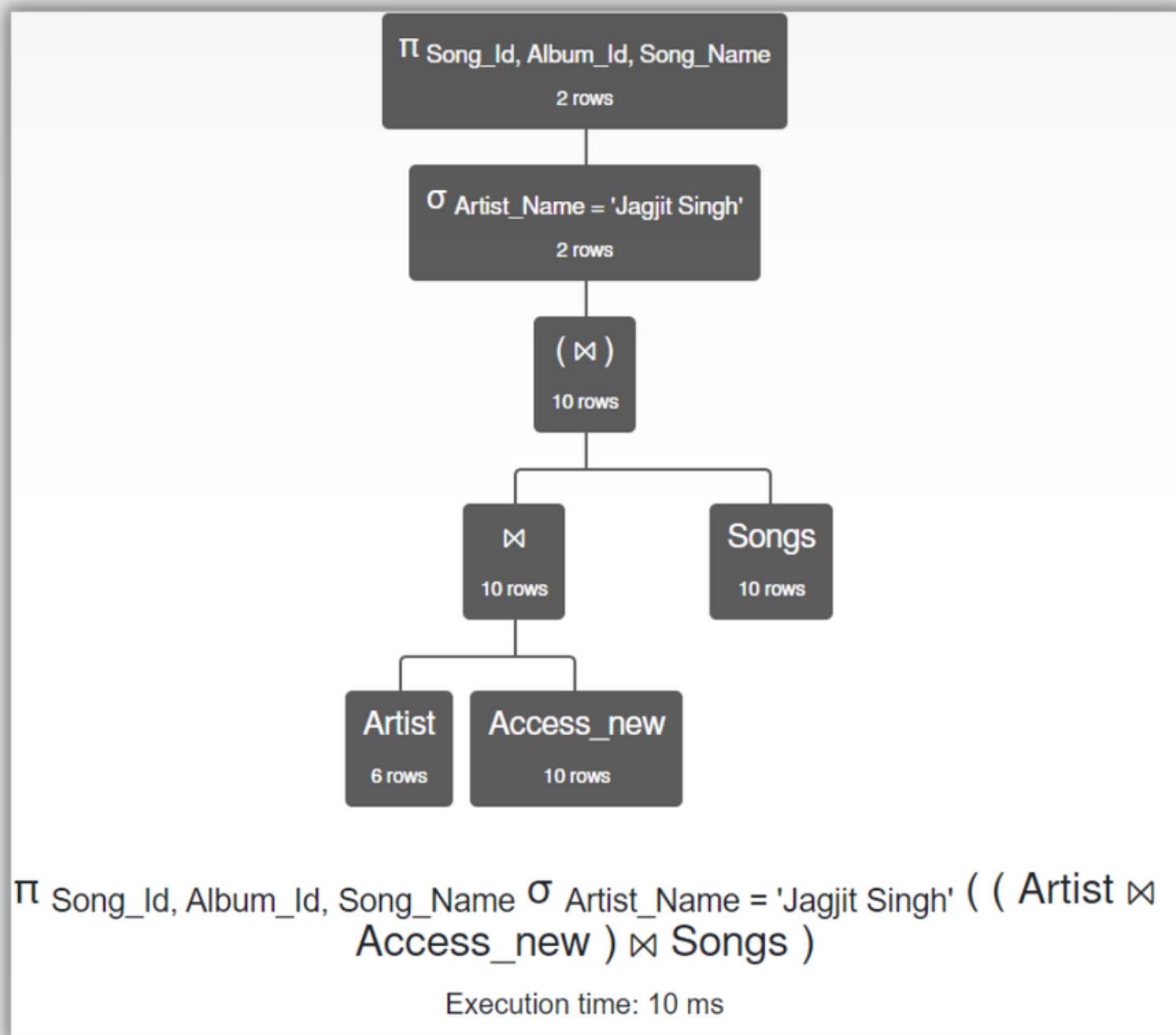
Input

- Artists : (Artist_ID, Artist_Name)
- Songs: (Song_Name, Genre, Duration, Album_ID, Artist_ID, Movie, Song_ID)
- Access_new: (Song_id, artist_id, Album_Id)

Output

$$\pi \text{Song_Id, Album_Id, Song_Name } \sigma \text{Artist_Name} = \text{'Jagjit Singh'} (\text{Artists} \\ \text{Access_new} \text{ Songs})$$

Output



Access_new.Song_Id	Access_new.Album_Id	Songs.Song_Name
'208'	'56'	'Garaj Baras'
'209'	'56'	'Badi Nazuk'

Query 5:

- List the song_Name and Genre of Songs in Playlist P2.

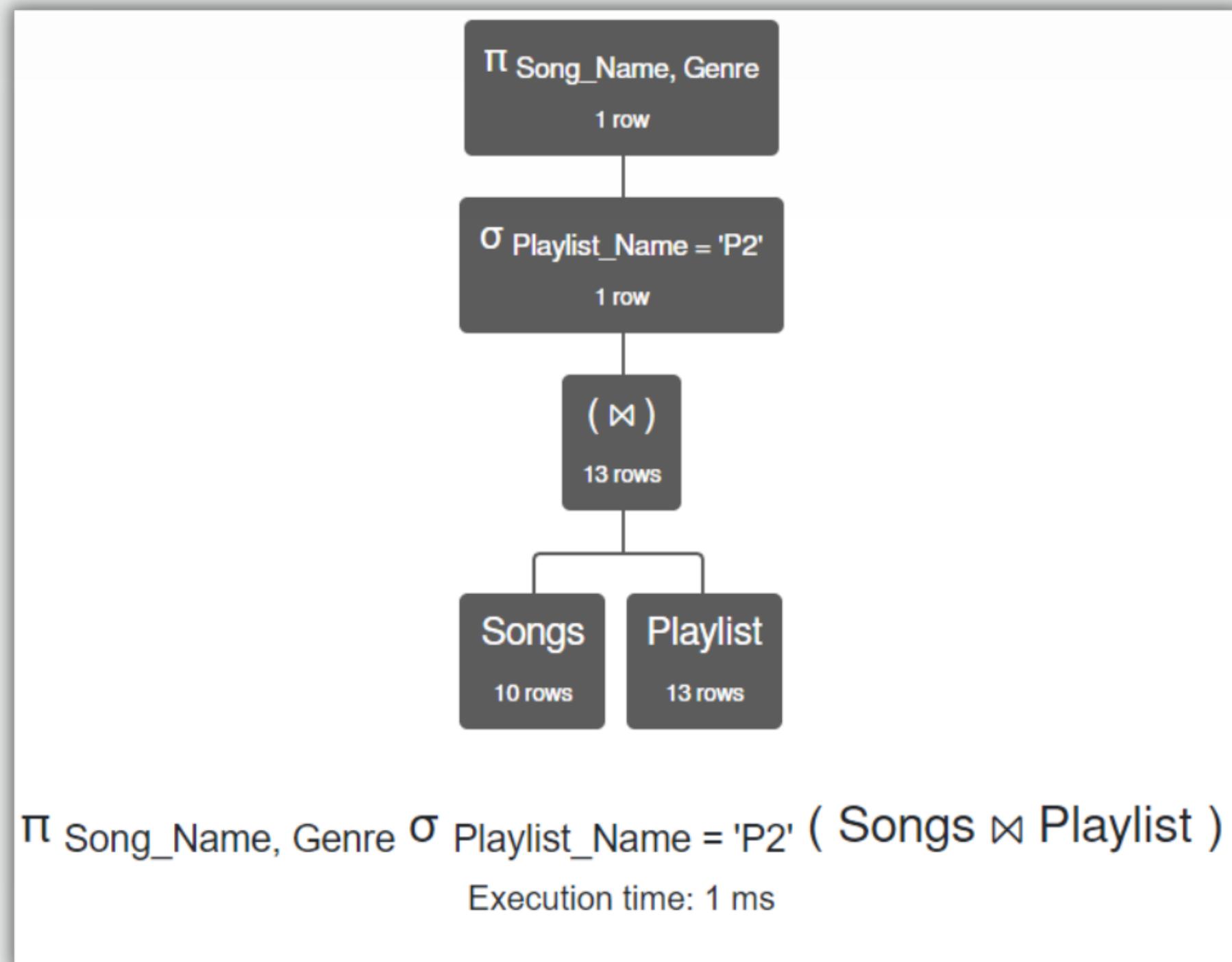
Input

- Songs (Song_Id, Genre, Song_Name, Movie, Duration)
- Playlists (playlist_Id, Song_Name, Playlist_Name)

Output

$$\pi \text{Song_Name, Genre } (\sigma \text{Playlist_Name} = \\ 'P2' (\text{Songs} \text{ } \text{Playlist}))$$

Output



Songs.Song_Name	Songs.Genre
'In Dino'	'Rock'

Thank
you!

