DBMS Project

Topic: Music Management System

By: Veerendra, Madhavan, Issac

CERTIFICATE

This is to certify that the DBMS Project titled “Music Management System" has been done by V S S Veerendra Kumar(422179), Karthikeya Madhavan(422134), D Issac(422135) pursuing Bachelor of Technology, Computer Science & Engineering during semester IV from National Institute of Technology Andhra Pradesh towards DBMS Lab.

Mr. D. Prasad
Faculty
CSE Department
NIT Andhra Pradesh

Table of Contents

ABSTRACT

- The Music Management System (MMS) is designed to streamline the organization and accessibility of music-related data within an online platform. It encompasses functionalities to manage albums, songs, artists, and customer interactions. Each album is uniquely identified by an Album ID and features attributes such as Title, Price, and Release Date. Songs, identified by Song ID, can belong to multiple albums and genres, each with attributes like Title and Play Time. Artists are represented by Artist IDs and include details like Name and Debut Date. Customers register with the system, providing essential information like Customer ID, Name, Address, Phone Number, and Birthday. Orders, identified by Order ID, include details such as Order Date, Total Price, Payment Method, and Delivery Option, facilitating seamless transactions. The MMS aims to optimize data management practices, enhance operational efficiency, and improve user experience within the music platform.

INTRODUCTION

- In the ever-evolving landscape of digital music consumption, effective management of music-related data is paramount for online platforms to thrive. This project introduces the Music Management System (MMS), a comprehensive solution designed to streamline the organization and accessibility of music-related information. With a focus on albums, songs, artists, and customer interactions, the MMS aims to revolutionize the way online music platforms operate. By centralizing and efficiently managing data pertaining to albums, songs, artists, and customers, the MMS promises to enhance operational efficiency, improve user experience, and ultimately elevate the success of music platforms in the digital age. This introduction provides an overview of the MMS project, highlighting its significance in the context of the evolving digital music industry and outlining the key objectives and functionalities it aims to achieve.

EXISTING SYSTEM AND ADVANTAGES

EXISTING SYSTEM

- Handling of transactions such as album purchases and customer orders through manual processes.
- Utilization of separate databases or spreadsheets for storing various types of data

ADVANTAGES

- Stability: Established processes provide stability.
- Minimal Disruption: Avoids potential downtime.
- Historical Data: Valuable for analysis.

PROPOSED SYSTEM AND DISADVANTAGES

PROPOSED SYSTEM

- Centralized Database: Single storage for all music data.
- Enhanced Efficiency: Streamlined operations.
- Improved Data Integrity: Ensuring accuracy.
- Scalability: Designed for future growth.

DISADVANTAGES

- Initial Cost: Setting up and automation require investment.
- Data Migration Challenges: Transitioning data poses risks.

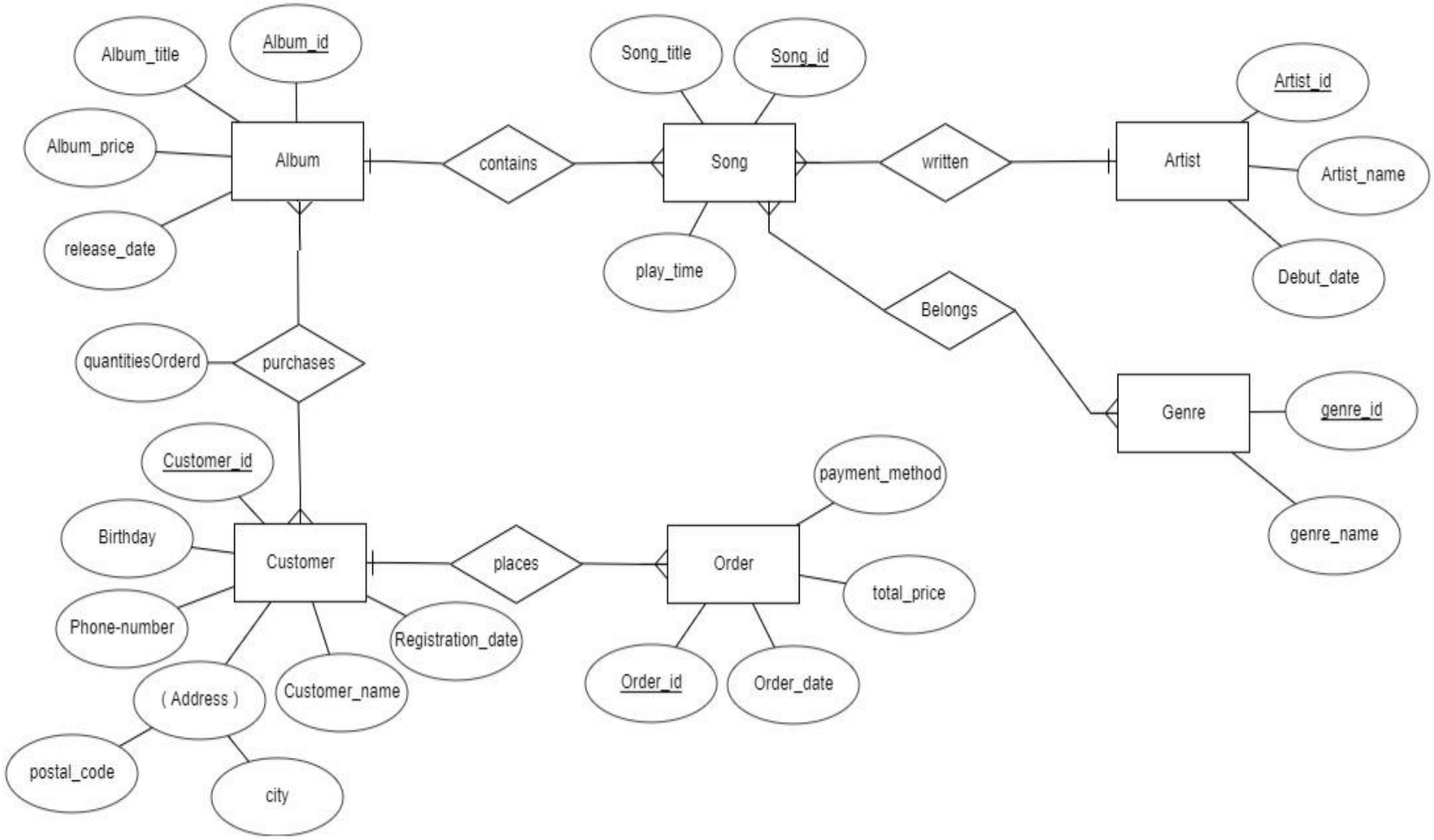
Entities & their Attributes

Entity Name	Attributes	Description
Album	Album_id, Album_title, Album_price, release_date	Collection of music tracks released together, identified by Album ID.
Song	Song_id, Song_title, play_time	Individual music piece with unique Song ID, including title, playtime, and genre.
Artist	Artist_id, Artist_name, Debut_date	Creator of music, identified by Artist ID, with attributes like name and debut date.
Genre	genre_id, genre_name	Category or style of music that songs can belong to, defining their musical characteristics.
Customer	Customer_id, Birthday, Phone_Number, Address, Customer_name, Registration_date	Platform member providing personal details like name, address, and contact information.
Order	Order_id, Order_date, total_price, payment_method	Transaction initiated by a customer to purchase albums, identified by Order ID, including order details and payment information.

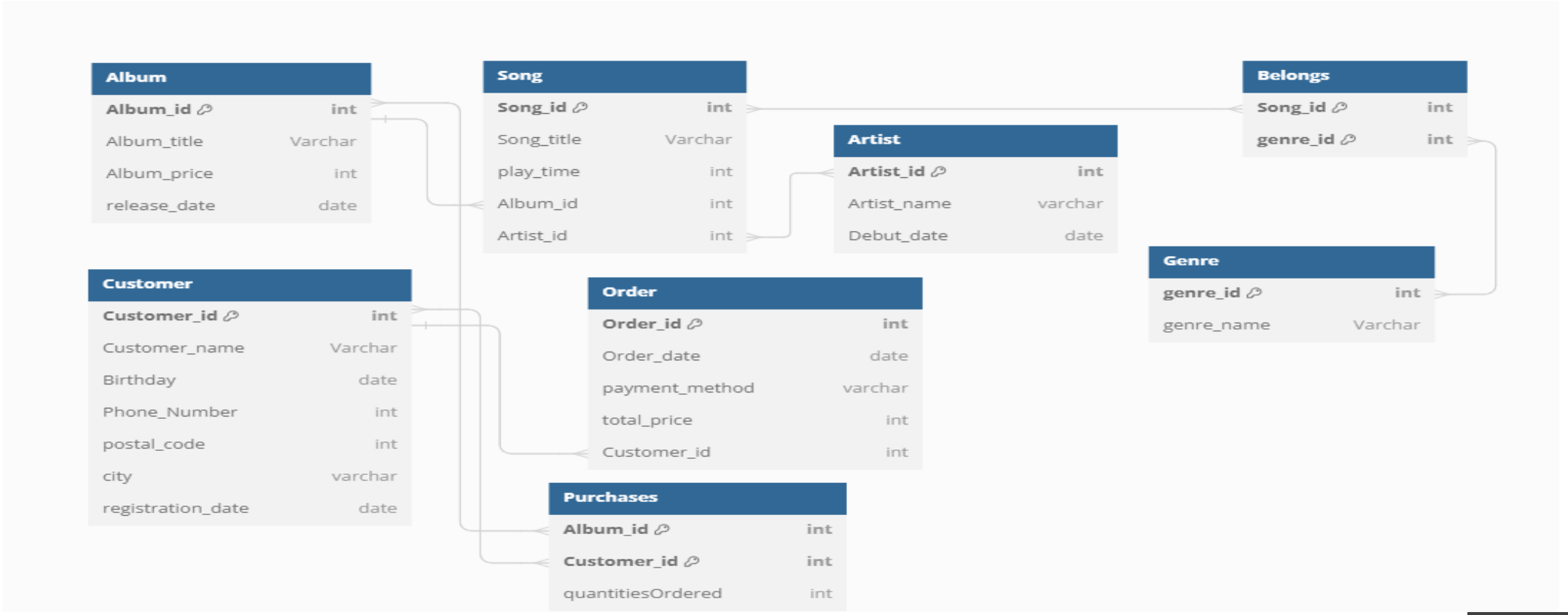
Relationship & Cardinalities

Entities	Relationship	Cardinality	Explanation
Album & Song	contains	One to Many	An album can contain multiple songs
Song & Artist	written	Many to One	A song can be written by multiple artists, and an artist can write multiple songs
Song & Genre	belongs	Many to Many	A song can belong to multiple genres, and a genre can have multiple songs
Album & Customer	purchases	Many to Many	Multiple album can be purchased by multiple customers
Customer & Order	places	One to Many	A customer can place multiple orders

ER Diagram


Album_title
Album_id
Album_price
Album
release_date
contains
Song_title
Song_id
Song
play_time
written
Artist_id
Artist
Artist_name
Debut_date
Belongs
quantitiesOrderd
purchases
Genre
genre_id
genre_name
Customer_id
Birthday
Customer
Phone-number
(Address)
postal_code
city
Customer_name
Registration_date
places
payment_method
Order
total_price
Order_id
Order_date

Relational Schema Before Normalization



Schema Refinement(Normalization)

Album: Album_id, Album_title, Album_price, release_date

F. D = {Album_id -> Album_title, Album_price, release_date}

- Prime attributes = {Album_id } Non-Prime attributes = {Album_title, Album_price, release_date}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute.

- 2NF

Non prime attributes are fully dependent on prime attributes

- 3NF

Non prime attributes are not determining other non prime attributes

Chosen Primary Key: Album_id

Schema Refinement(Normalization)

Song: Song_id, Song_title, play_time

- F. D = {Song_id -> Song_title, play_time}
- Prime attributes = {Song_id } Non-Prime attributes = {Song_title, play_time}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute.

- 2NF

Non prime attributes are fully dependent on prime attributes

- 3NF

Non prime attributes are not determining other non prime attributes

Chosen Primary Key: Song_id

Schema Refinement(Normalization)

Artist: Artist_id, Artist_name, Debut_date

F. D = {Artist_id -> Artist_name, Debut_date}

- Prime attributes = {Artist_id} Non-Prime attributes = {Artist_name, Debut_date}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Non prime attributes are fully dependent on prime attributes, i.e. No Partial Dependency

- 3NF

Non prime attributes are not determining other non prime attributes i.e. No Transitive Dependency

Chosen Primary Key: Artist_id

Schema Refinement(Normalization)

Belongs: Song_id, genre_id

-Prime attributes = {(Song_id, genre_id)}

- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Non prime attributes are fully dependent on prime attributes, i.e. No Partial Dependency

- 3NF

It is in 3NF because no non-prime attribute determines a non-prime attribute, i.e., no Transitive Dependency

Chosen Primary Key: (Song_id, genre_id)

Schema Refinement(Normalization)

Genre: genre_id, genre_name

- F.D = {genre_id -> genre_name}
- Prime attributes = {genre_id } Non-Prime attributes = {genre_name}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Since there are no non prime attributes there is no Partial Dependency

- 3NF

Since there are no non prime attributes there is no Tranisitive Dependency

Chosen Primary Key: genre_id

Schema Refinement(Normalization)

Customer: Customer_id, Customer_name, Birthday, Phone_Number, postal_code, city, registration_date

- F.D = {Customer_id -> Customer_name, Birthday, Phone_Number, postal_code, city, registration_date

  postal_code-> city}
- Prime attributes = {Customer_id } Non-Prime attributes = {Customer_name, Birthday, Phone_Number, postal_code, city, registration_date}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Non prime attributes are fully dependent on prime attributes, i.e. No Partial Dependency

- 3NF

Non prime attributes are determining other non prime attributes i.e. Transitive Dependency {postal_code-> city}

Chosen Primary Key: Customer_id

To make Customer table in 3NF decompose Customer into Customer and Address Tables

Schema Refinement(Normalization)

Address: postal_code, city

F.D = {postal_code -> city}

- Prime attributes ={ postal_code} Non-Prime attributes = {city}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Since there are no non prime attributes there is no Partial Dependency

- 3NF

Since there are no non prime attributes there is no Tranisitive Dependency

Chosen Primary Key: postal_code

Schema Refinement(Normalization)

Customer: Customer_id, Customer_name, Birthday, Phone_Number, postal_code, registration_date

- F.D = {Customer_id -> Customer_name, Birthday, Phone_Number, postal_code, registration_date }
- Prime attributes = {Customer_id } Non-Prime attributes = {Customer_name, Birthday, Phone_Number, postal_code, registration_date}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Non prime attributes are fully dependent on prime attributes, i.e. No Partial Dependency

- 3NF

Non prime attributes are not determining other non prime attributes i.e. NO Transitive Dependency

Chosen Primary Key: Customer_id

Schema Refinement(Normalization)

Purchases: Album_id, Customer_id, quantitiesOrdered

F.D = { (Album_id, Customer_id) -> quantitiesOrdered}

- Prime attributes = { (Album_id, Customer_id)} Non-Prime attributes = {quantitiesOrdered}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

Non prime attributes are fully dependent on prime attributes, i.e. No Partial Dependency

- 3NF

Non prime attributes are not determining other non prime attributes i.e. No Transitive Dependency

Chosen Primary Key: Album_id, Customer_id

Schema Refinement(Normalization)

Orders: Order_id, Order_date, payment_method, total_price

F.D = {Order_id -> Order_date, payment_method, total_price}

- Prime attributes = {Order_id } Non-Prime attributes = {Order_date, payment_method, total_price}
- 1NF

The table is already in 1NF as there are no multiple values in any attribute, i.e. attributes are atomic

- 2NF

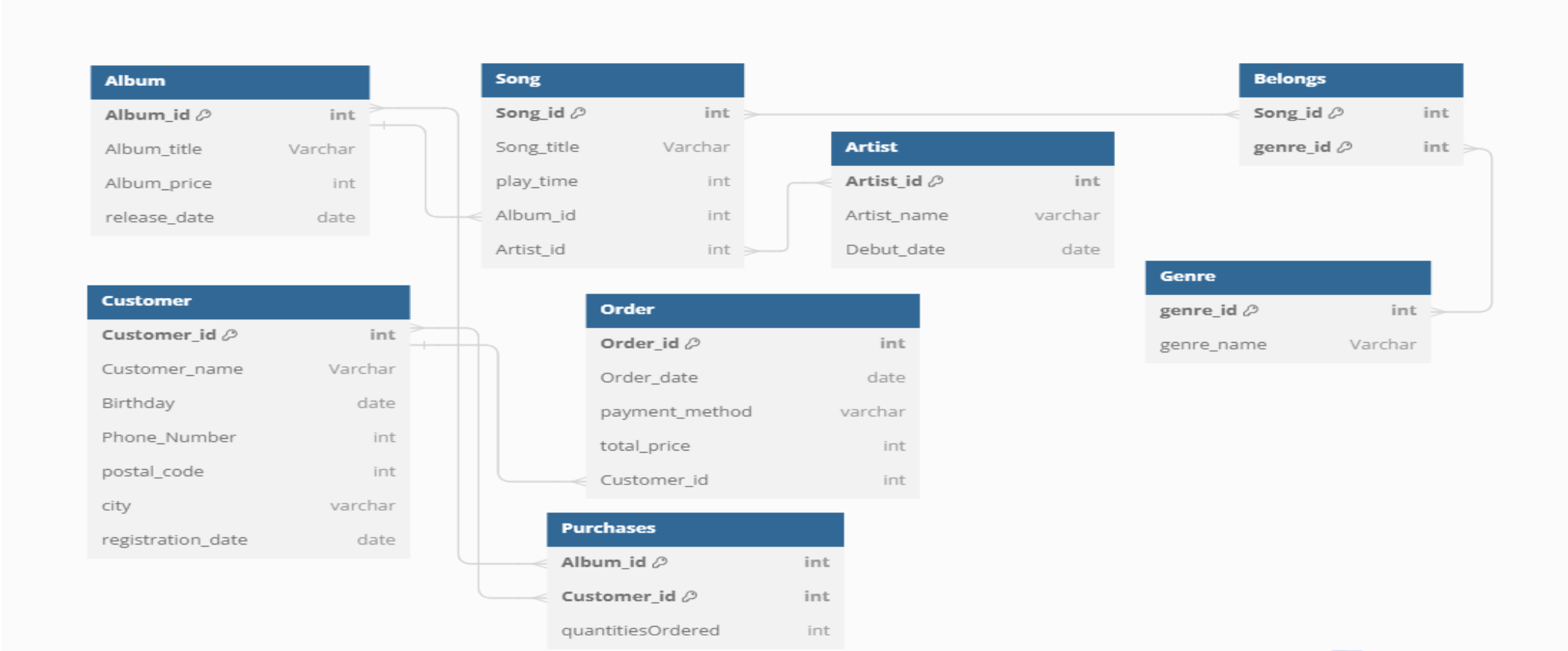
Non prime attributes are fully dependent on prime attributes, i.e. No Partial Dependency

- 3NF

Non prime attributes are not determining other non prime attributes i.e. No Transitive Dependency

Chosen Primary Key: Order_id

Relational Schema after Normalization



INTEGRITY CONSTRAINTS

- Album_id int primary key not null,
- Album_title Varchar(20) not null,
- Song_title Varchar(20) not null,
- play_time int not null,
- Customer_name Varchar(20) not null,
- Phone_Number Varchar(20) check(length(Phone_Number)=10),

Schema Creation in MySQL

```
Create databasecreate database musicDB;
use musicDB;
Create Album table
create table Album (
        Album_id int primary key not null,
        Album_title Varchar(20) not null,
        Album_price int,
        release_date date);
```

```
-- Create Artist table
create table Artist (
        Artist_id int primary key not null,
        Artist_name Varchar(20) not null,
        Debut_date date);
-- Create Song table
create table Song (
        Song_id int primary key not null,
        Song_title Varchar(20) not null,
        play_time int not null,
        Album_id int,
        Artist_id int,
        foreign key (Album_id) REFERENCES Album(Album_id),
        foreign key (Artist_id) REFERENCES Artist(Artist_id));
```

Continuation

```
 -- Create Genre table
create table Genre (
genre_id int primary key not null,
genre_name Varchar(20) not null);
Create Belongs table
create table Belongs (
genre_id int,
Song_id int,
foreign key (genre_id) REFERENCES Genre(genre_id),
foreign key (Song_id) REFERENCES Song(Song_id));
```

```
-- Create Customer table
create table Customer (
        Customer_id int primary key not null,
        Customer_name Varchar(20) not null,
        Birthday date,
        Phone_NumberVarchar(20)check(length(Phone_Number)=10),
        postal_code int);
ALTER TABLE Customer ADD INDEX idx_postal_code (postal_code);
```

Continuation

```
-- Create Address table
create table Address (
        postal_code int primary key,
        city Varchar(20) not null,
        FOREIGN KEY (postal_code) REFERENCES
        Customer(postal_code));
-- Create Purchases table
create table Purchases (
        Album_id int,
        Customer_id int,
        quantitiesOrdered int,
        foreign key (Album_id) REFERENCES
        Album(Album_id),
        foreign key (Customer_id) REFERENCES
        Customer(Customer_id));
```

```
-- Create Orders table
create table Orders (
        Order_id int primary key not null,
        Order_date date,
        payment_method Varchar(20),
        total_price int,
        Customer_id int,
        foreign key (Customer_id)  REFERENCES
        Customer(Customer_id));
```

Output Screenshots

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	23:39:25	DROP DATABASE `musicdb`	9 row(s) affected	0.188 sec
2	23:39:30	create database musicDB	1 row(s) affected	0.000 sec
3	23:39:30	use musicDB	0 row(s) affected	0.000 sec
4	23:39:30	create table Album (Album_id int primary key not null, Album_title Varchar(20) not null, Album_price int, r...	0 row(s) affected	0.016 sec
5	23:39:30	create table Artist (Artist_id int primary key not null, Artist_name Varchar(20) not null, Debut_date date)	0 row(s) affected	0.016 sec
6	23:39:30	create table Song (Song_id int primary key not null, Song_title Varchar(20) not null, play_time int not null, ...	0 row(s) affected	0.109 sec
7	23:39:30	create table Genre (genre_id int primary key not null, genre_name Varchar(20) not null)	0 row(s) affected	0.016 sec
8	23:39:30	create table Belongs (genre_id int, Song_id int, foreign key (genre_id) REFERENCES Genre(genre_id), for...	0 row(s) affected	0.047 sec
9	23:39:30	create table Customer (Customer_id int primary key not null, Customer_name Varchar(20) not null, Birthday ...	0 row(s) affected	0.031 sec
10	23:39:30	ALTER TABLE Customer ADD INDEX idx_postal_code (postal_code)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
11	23:39:31	create table Address (postal_code int primary key, city Varchar(20) not null, FOREIGN KEY (postal_code) ...	0 row(s) affected	0.032 sec
12	23:39:31	create table Purchases (Album_id int, Customer_id int, quantitiesOrdered int, foreign key (Album_id) RE...	0 row(s) affected	0.078 sec
13	23:39:31	create table Orders (Order_id int primary key not null, Order_date date, payment_method Varchar(20), to...	0 row(s) affected	0.062 sec

Output Screenshots

Output

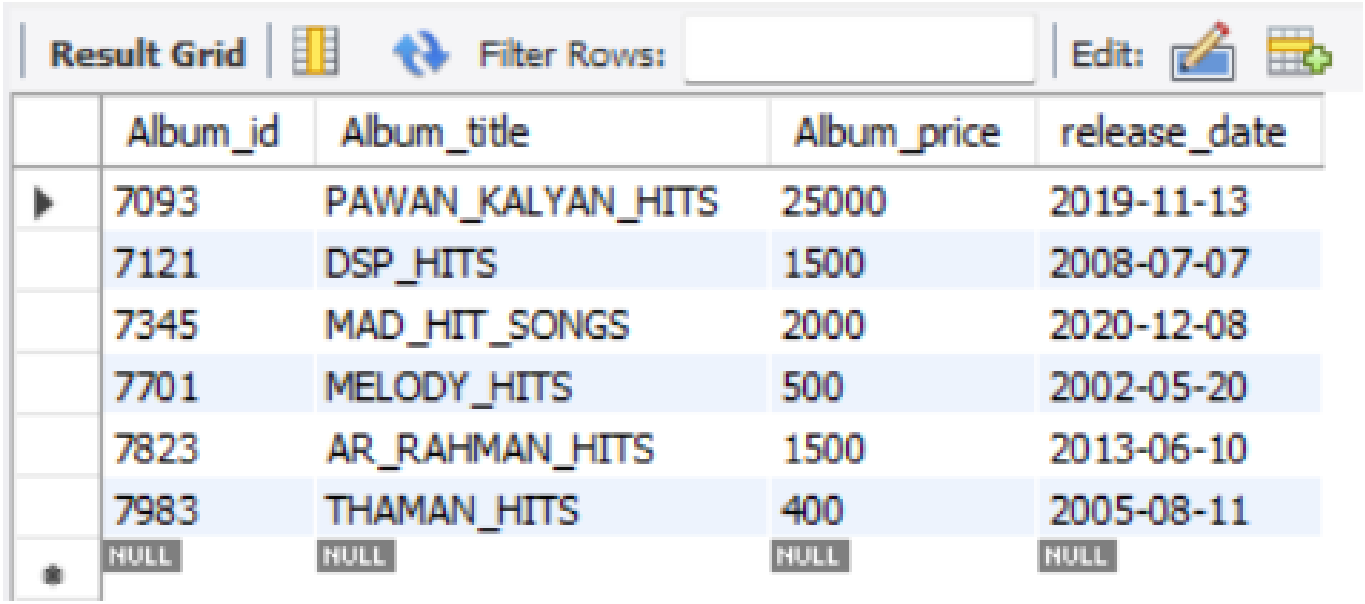
Action Output

#	Time	Action	Message	Duration / Fetch
1	03:37:30	ALTER TABLE Diet_Plan MODIFY COLUMN dietplan_id INT NOT NULL, MODIFY COLUMN dietplanname VA...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.062 sec
2	03:37:42	ALTER TABLE Customer MODIFY user_id INT NOT NULL, MODIFY dietplan_id INT NOT NULL, MODIFY dob ...	Error Code: 1832. Cannot change column 'user_id': used in a foreign key constraint 'customer_ibfk_1'	0.000 sec
3	03:38:12	ALTER TABLE Customer MODIFY dob DATE NOT NULL, MODIFY height FLOAT NOT NULL, MODIFY weight...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
4	03:38:19	ALTER TABLE Nutritionist ADD CHECK (specialization <> ''), ADD CHECK (gender IN ('Male', 'Female', 'Other'))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.062 sec
5	03:38:34	ALTER TABLE Meal ADD CHECK (dayperiod IN ('breakfast', 'lunch', 'dinner')), MODIFY mealname VARCHAR(2...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.047 sec
6	03:38:42	ALTER TABLE Food_Item ADD CHECK (calories >= 0 AND carbohydrates >= 0 AND proteins >= 0 AND fats >=...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.062 sec
7	03:38:47	ALTER TABLE Meal_Item ADD CHECK (quantity >= 0), ADD UNIQUE (mealitem_id)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.078 sec
8	03:38:59	ALTER TABLE Food_Item_Vitamin MODIFY vitamin VARCHAR(255) NOT NULL, ADD UNIQUE (fooditem_id, vi...	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.062 sec

Insert sample values into Album table

```
insert into Album (Album_id,Album_title,Album_price,release_date)
values(7701,"MELODY_HITS",500,"2002-05-20"),
        (7823,"AR_RAHMAN_HITS",1500,"2013-06-10"),
         (7093,"PAWAN_KALYAN_HITS", 25000, "2019-11-13"),
         (7983, "THAMAN_HITS",400, "2005-08-11"),
         (7121, "DSP_HITS" ,1500 ,"2008-07-07"),
         (7345, "MAD_HIT_SONGS", 2000,"2020-12-08");
```

select * from Album;



Result Grid | Filter Rows: | Edit:

Album_id	Album_title	Album_price	release_date
7093	PAWAN_KALYAN_HITS	25000	2019-11-13
7121	DSP_HITS	1500	2008-07-07
7345	MAD_HIT_SONGS	2000	2020-12-08
7701	MELODY_HITS	500	2002-05-20
7823	AR_RAHMAN_HITS	1500	2013-06-10
7983	THAMAN_HITS	400	2005-08-11
NULL	NULL	NULL	NULL

Insert sample values into Artist table

```
insert into Artist (Artist_id,Artist_name,Debut_date) values
                (8769,"BALASUBRAMANYAM","1981-05-20"),
                 (8469,"AR_RAHMAN","1982-06-10"),
                 (8596,"ANIRUDH","1981-11-13"),
                 (8367, "THAMAN","1982-08-11"),
                 (8473, "DSP","1982-07-07"),
                 (8639, "BHEEM","1981-12-08");
```

select * from Artist;

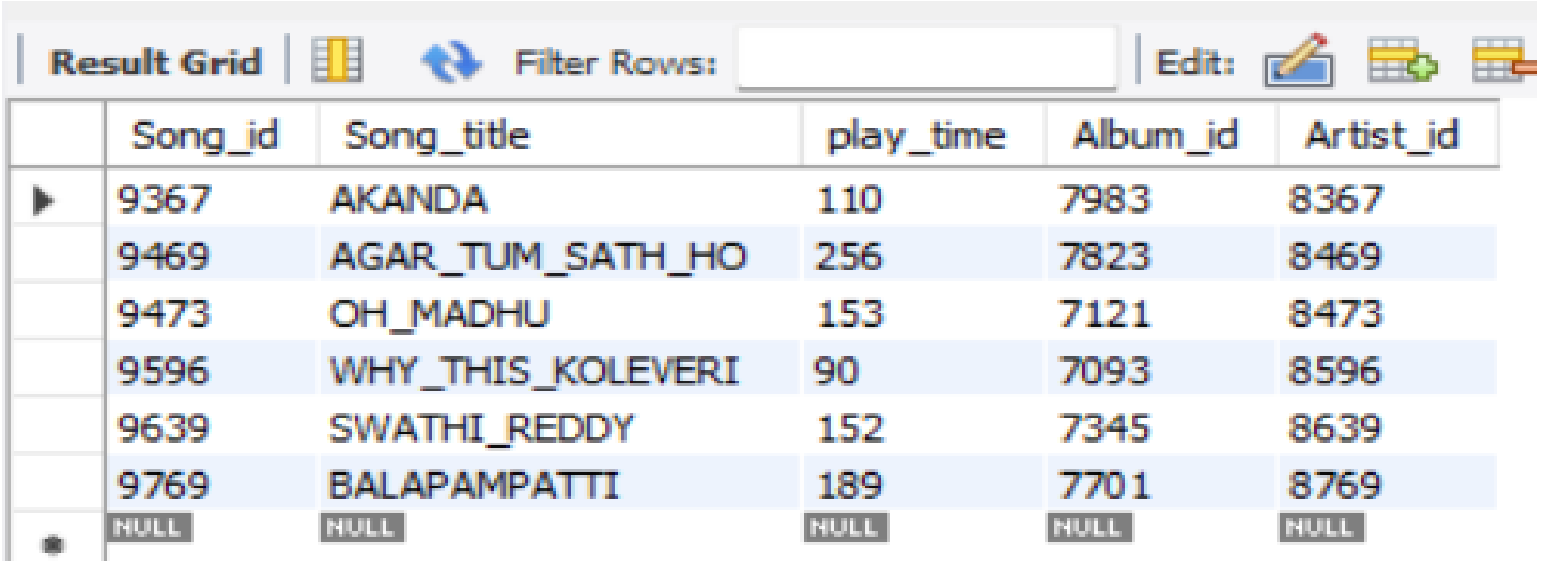
Result Grid | Filter Rows:

Artist_id	Artist_name	Debut_date
8367	THAMAN	1982-08-11
8469	AR_RAHMAN	1982-06-10
8473	DSP	1982-07-07
8596	ANIRUDH	1981-11-13
8639	BHEEM	1981-12-08
8769	BALASUBRAMANYAM	1981-05-20
NULL	NULL	NULL

Insert sample values into Song table

```
insert into Song (Song_id,Song_title,play_time,Album_id,Artist_id)values
        (9769,"BALAPAMPATTI",189,7701,8769),
        (9469,"AGAR_TUM_SATH_HO",256,7823, 8469),
        (9596,"WHY_THIS_KOLEVERI", 90, 7093, 8596),
        (9367, "AKANDA",110,7983, 8367),
        (9473, "OH_MADHU" ,153,7121, 8473),
        (9639, "SWATHI_REDDY",152,7345, 8639);
```

select * from Song;



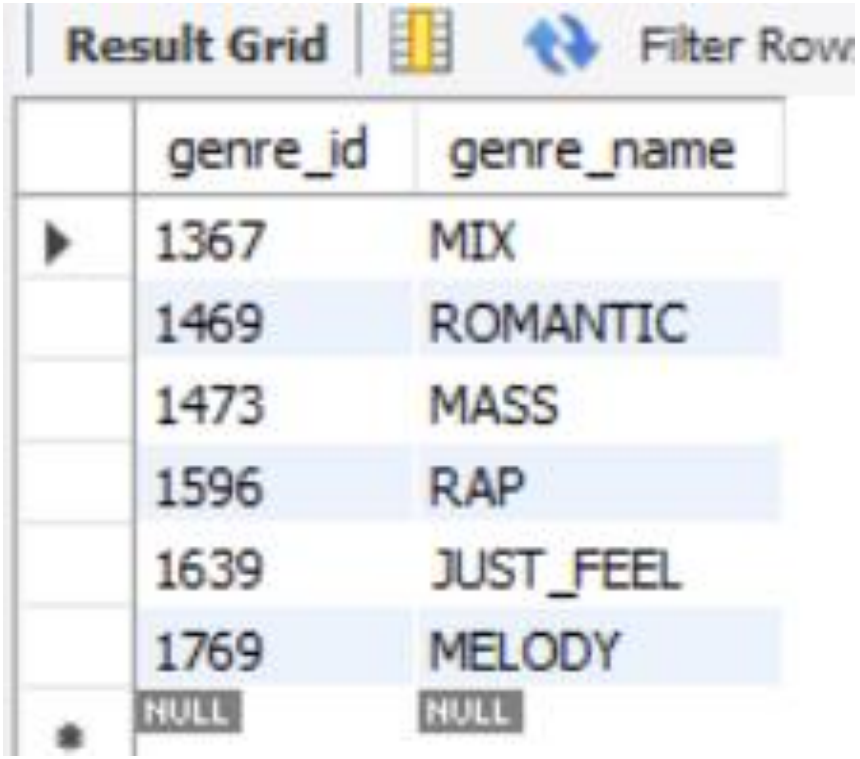
Result Grid | Filter Rows: | Edit:

Song_id	Song_title	play_time	Album_id	Artist_id
9367	AKANDA	110	7983	8367
9469	AGAR_TUM_SATH_HO	256	7823	8469
9473	OH_MADHU	153	7121	8473
9596	WHY_THIS_KOLEVERI	90	7093	8596
9639	SWATHI_REDDY	152	7345	8639
9769	BALAPAMPATTI	189	7701	8769
NULL	NULL	NULL	NULL	NULL

Insert sample values into Genre table

```
insert into Genre (genre_id,genre_name) values
        (1769,"MELODY"),
        (1469,"ROMANTIC"),
        (1596,"RAP"),
        (1367, "MIX"),
        (1473, "MASS"),
        (1639, "JUST_FEEL");
```

select * from Genre;

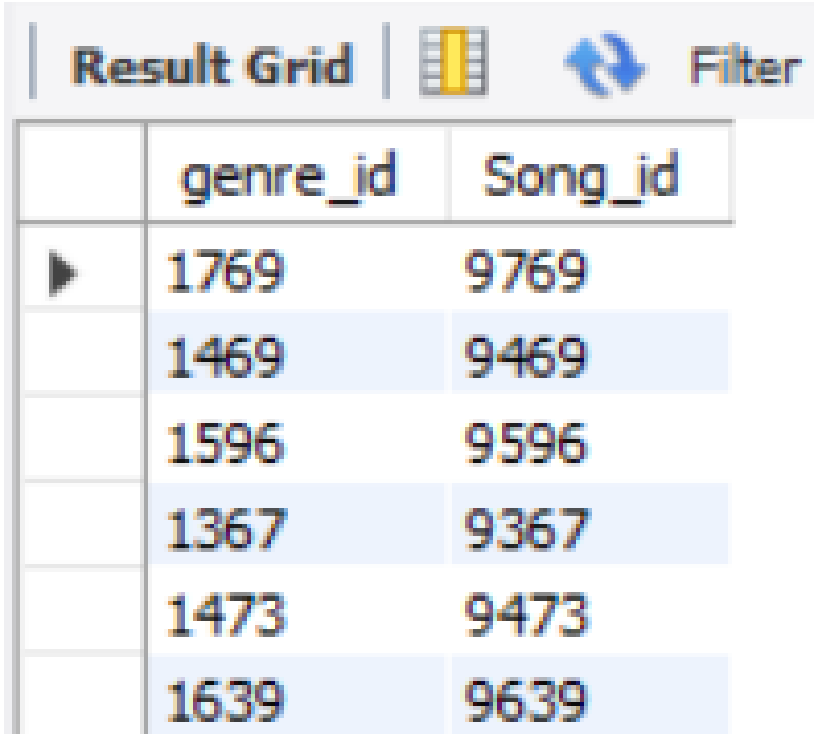


genre_id	genre_name
1367	MIX
1469	ROMANTIC
1473	MASS
1596	RAP
1639	JUST_FEEL
1769	MELODY
NULL	NULL

Insert sample values into Belongs table

```
insert into Belongs (genre_id,Song_id) values
        (1769,9769),
        (1469,9469),
        (1596,9596),
        (1367,9367),
        (1473,9473),
        (1639,9639);
```

select * from Belongs;



	genre_id	Song_id
▶	1769	9769
	1469	9469
	1596	9596
	1367	9367
	1473	9473
	1639	9639

Insert sample values into Customer table

```
insert into Customer
(Customer_id,Customer_name,Birthday,Phone_Number,postal_code)values
(3769,"MADHAVAN","2004-03-21","4567890123",534101),
        (3469,"VEERENDRA","2005-04-07","5678901234",533342),
        (3596,"ISSAC", "2004-08-01","7890123456",502355),
        (3367, "TEJA","2003-09-07","8901234567",532201),
        (3473, "SRINIVAS","1986-09-25" ,"9012345678",531001),
        (3639, "CHANDU","1999-10-17","3456789012",535001);
```

select * from Customer;

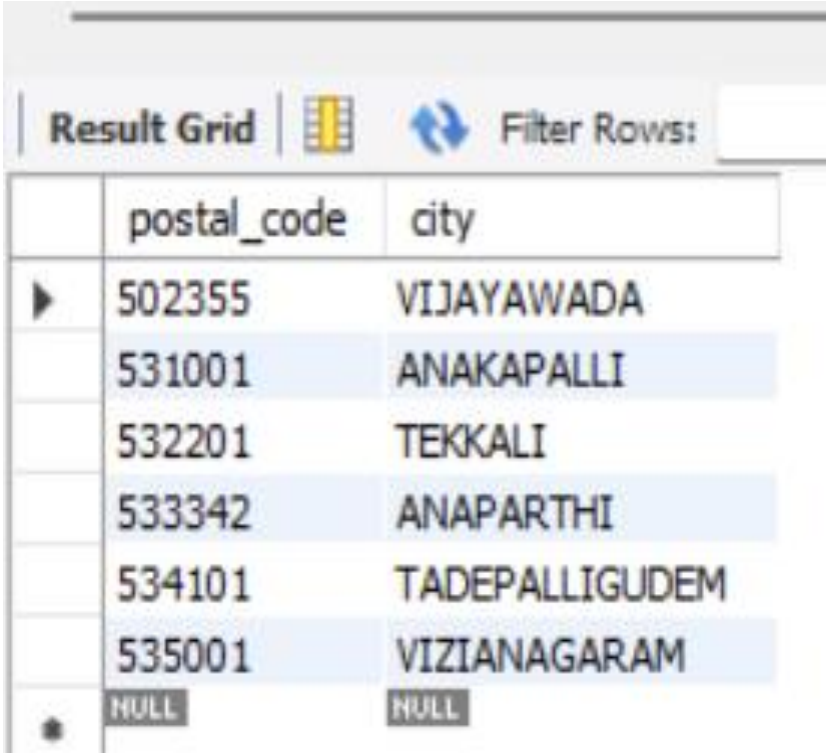
Result Grid | Filter Rows: | Edit: | Export/I

Customer_id	Customer_name	Birthday	Phone_Number	postal_code
3367	TEJA	2003-09-07	8901234567	532201
3469	VEERENDRA	2005-04-07	5678901234	533342
3473	SRINIVAS	1986-09-25	9012345678	531001
3596	ISSAC	2004-08-01	7890123456	502355
3639	CHANDU	1999-10-17	3456789012	535001
3769	MADHAVAN	2004-03-21	4567890123	534101
NULL	NULL	NULL	NULL	NULL

Insert sample values into Address table

```
insert into Address (postal_code,city)values(534101,"TADEPALLIGUDEM"),
                                             (533342,"ANAPARTHI"),
                                             (502355,"VIJAYAWADA"),
                                             (532201, "TEKKALI"),
                                             (531001, "ANAKAPALLI"),
                                             (535001, "VIZIANAGARAM");
```

select * from Address;



postal_code	city
502355	VIJAYAWADA
531001	ANAKAPALLI
532201	TEKKALI
533342	ANAPARTHI
534101	TADEPALLIGUDEM
535001	VIZIANAGARAM
NULL	NULL

Insert sample values into Purchases table

```
insert into Purchases (Album_id,Customer_id,quantitiesOrdered)values(7701,3769,5),
                                                                    (7823,3469,2),
                                                                    (7093,3596,0),
                                                                    (7983,3367,6),
                                                                    (7121,3473,10),
                                                                    (7345,3639,6);
```

select * from Purchases;

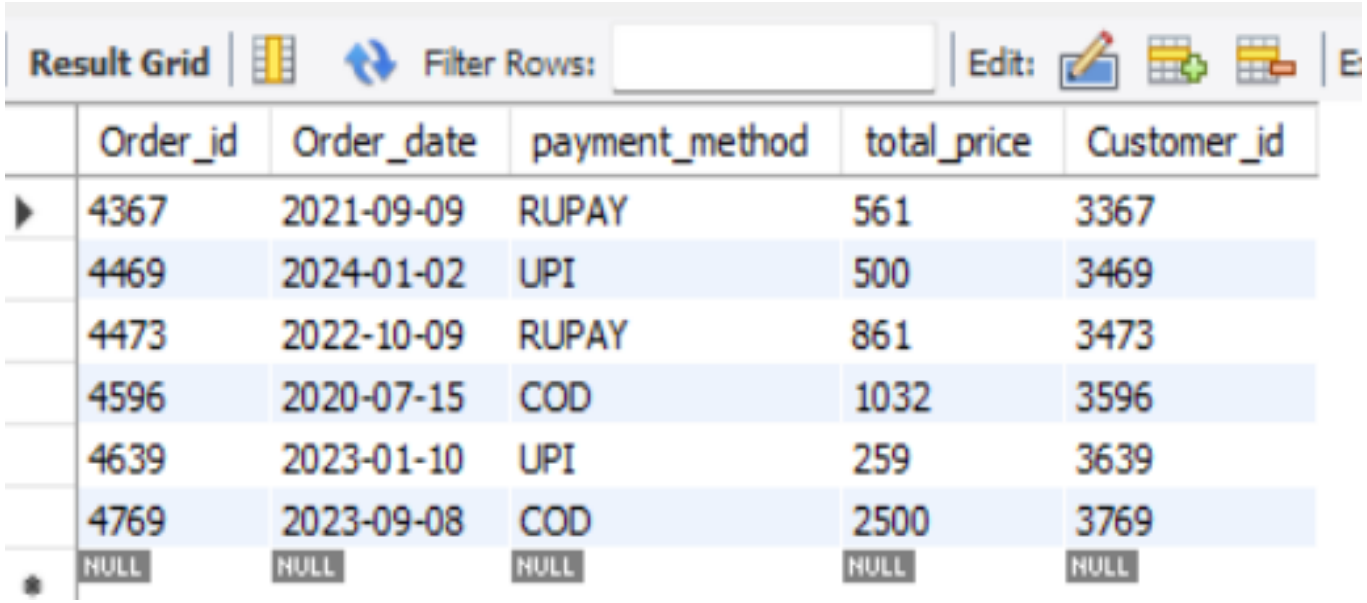
Result Grid | Filter Rows:

Album_id	Customer_id	quantitiesOrdered
7701	3769	5
7823	3469	2
7093	3596	0
7983	3367	6
7121	3473	10
7345	3639	6

Insert sample values into Orders table

```
insert into Orders (Order_id,Order_date,payment_method,total_price,Customer_id)values
        (4769,"2023-09-08","COD",2500,3769),
         (4469,"2024-01-02","UPI",500, 3469),
         (4596,"2020-07-15", "COD", 1032, 3596),
         (4367, "2021-09-09","RUPAY",561, 3367),
         (4473, "2022-10-09" ,"RUPAY",861, 3473),
         (4639, "2023-01-10","UPI",259, 3639);
```

select * from Orders;



Order_id	Order_date	payment_method	total_price	Customer_id
4367	2021-09-09	RUPAY	561	3367
4469	2024-01-02	UPI	500	3469
4473	2022-10-09	RUPAY	861	3473
4596	2020-07-15	COD	1032	3596
4639	2023-01-10	UPI	259	3639
4769	2023-09-08	COD	2500	3769
NULL	NULL	NULL	NULL	NULL

CONCLUSION

- In conclusion, the implementation of a music management system offers significant advantages for S Record and its operations. By centralizing data and automating processes, the system enhances efficiency, improves data integrity, and provides valuable insights through analytics. Additionally, personalized user experiences and robust security measures contribute to customer satisfaction and trust in the platform. With scalability to accommodate growth and adaptability to evolving industry needs, the music management system positions S Record for long-term success in the competitive online music market. Through continuous refinement and innovation, S Record can leverage its enhanced data management practices to drive business growth, foster artist collaborations, and deliver exceptional music experiences to its customers.

Thank You