ONLINE MUSIC STREAMING PLATFORM USING DATABASE

PROJECT REPORT

Submitted by

SHOAIB AKHTAR (RA2211029010006)

SAHIRA (RA2211029010004)

TINA KASHYAP (RA2211028010226)

Under the guidance of

Dr P. Mahalakshmi

Assistant Professor, Department of Networking and Communications

In partial satisfaction of the requirements for the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

with specialization in Computer Networking

DEPARTMENT OF NETWORKING AND COMMUNICATION

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603 203

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603 203

BONAFIDE CERTIFICATE

Certified that this Project Report titled "**ONLINE MUSIC STREAMING PLATFORM USING DATABASE**" is the bonafide work done by:

SHOAIB AKHTAR (RA2211029010006)

SAHIRA (RA2211029010004)

TINA KASHYAP (RA2211028010226)

who completed the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

SIGNATURE

Dr P. Mahalakshmi

DBMS-Course Faculty

Assistant Professor

Department of Networking and Communications

SIGNATURE

Dr Annapurani Panaiyappan

Head of the Department

Department of Networking and Communications

SRMIST

SRMIST

Department of Networking and Communications

SRM Institute of Science and Technology

Own Work Declaration Form

Degree/Course: B. Tech in Computer Science and Engineering with specialization in Computer Networking / Cloud Computing

Names of the Students: Shoaib Akhtar, Sahira and Tina Kashyap

Registration Numbers: RA2211029010006, RA2211029010004, RA2211028010226

Title of Work: Online Music Streaming Platform using Database

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism, as listed in the University Website, Regulations, and the Education Committee guidelines.
We confirm that all the work contained in this assessment is our own except where indicated, and that we have met the following conditions:

- Clearly references / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc.)
- Given the sources of all pictures, data etc. that are not our own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (eg. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the course handbook/University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION
I am aware of and understand the University's policy on Academic misconduct and plagiarism and I certify that this assessment is my/our own work, except where indicated by referring, and that I have followed the good academic practices noted above.
If you are working in a group, please write your registration numbers and sign with the date for every student in your group.

ACKNOWLEDGMENT

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T. V. Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department, **Dr Annapurani Panaiyappan,** Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We register our immeasurable thanks to our Faculty Advisor, **Dr S. Ushasukhanya**, Assistant Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to our guide, **Dr P. Mahalakshmi**, Associate Professor, Department of Networking and Communications, SRM Institute of Science and Technology, for providing us with an opportunity to pursue our project under his mentorship. She provided us with the freedom and support to explore the research topics of our interest. Her passion for solving problems and helping us solve our problems has always been inspiring.

We sincerely thank **the staff, faculty and the students of the Department of Networking and Communications**, SRM Institute of Science and Technology, for their help during our project.

Finally, we would like to thank **our parents, family members, and friends** for their unconditional love, constant support, and encouragement.

TABLE OF CONTENTS

ABSTRACT

In today's digital era, university students encounter numerous challenges when accessing free online music streaming platforms. These challenges, ranging from advertisement overload to limited content availability and compromised audio quality, hinder students' ability to fully harness the benefits of music in their academic and personal lives. To address these issues, BeatFlow, a specialized music streaming platform for universities, emerges as a tailored solution. By offering an ad-free listening experience, a diverse library of music, high-fidelity audio, and flexible accessibility, BeatFlow aims to enhance the university music experience. This abstract explores the features and benefits of BeatFlow in alleviating the challenges faced by university students in accessing free online music streaming platforms, ultimately empowering students to unlock their full potential both academically and personally.

INTRODUCTION

In the bustling world of higher education, where students are constantly striving to strike a balance between academic rigor and personal well-being, music serves as a ubiquitous companion, offering solace, motivation, and inspiration. Free online music streaming platforms have emerged as indispensable tools for university students, providing access to an extensive catalog of songs and playlists to accompany their academic pursuits. However, amid the convenience and accessibility of these platforms, students encounter a myriad of challenges that hinder their ability to fully enjoy the benefits of music.

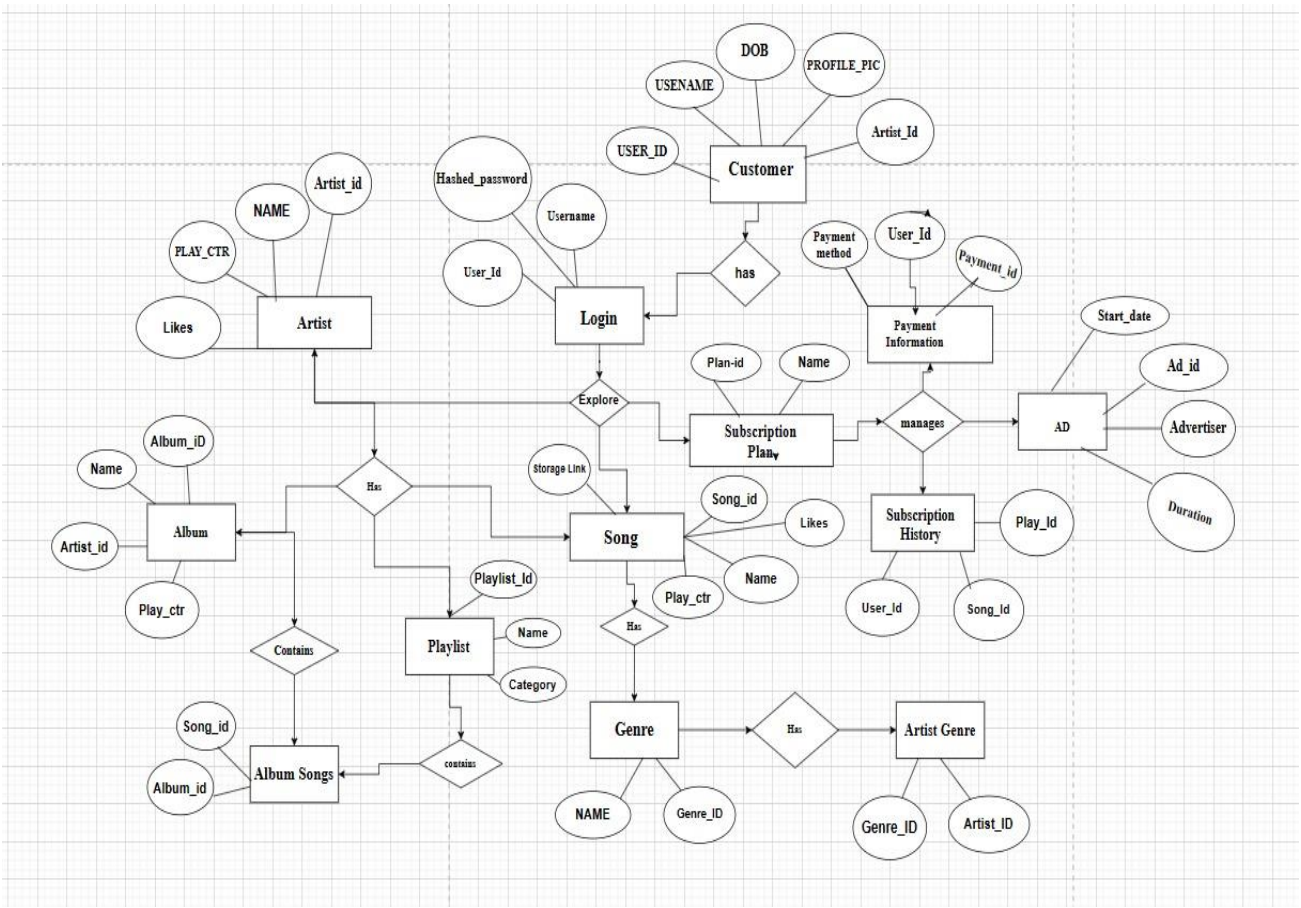
From the relentless intrusion of advertisements disrupting study sessions to the limitations in content availability and compromised audio quality, university students grapple with obstacles that detract from their music streaming experience. These challenges not only undermine students' academic performance but also impact their overall well-being, hindering their ability to cultivate a conducive study environment and find respite amidst the rigors of university life.

Recognizing the pressing need to address these challenges, BeatFlow emerges as a dedicated solution tailored to the unique needs of university students. By offering an ad-free listening experience, a diverse library of music spanning various genres and artists, high-fidelity audio, and flexible accessibility, BeatFlow endeavors to enhance the university music experience. This introduction sets the stage for an exploration of BeatFlow's features and benefits in mitigating the challenges faced by university students in accessing free online music streaming platforms, ultimately empowering them to harness the transformative power of music in their academic and personal lives.

LITERATURE SURVEY

A literature survey on online music streaming platforms entails a comprehensive investigation into the multifaceted landscape of this rapidly evolving industry. It begins by dissecting the technological underpinnings that power these platforms, scrutinizing the intricate infrastructure responsible for content delivery, encoding methods optimizing audio quality and bandwidth efficiency, and the sophisticated algorithms that drive content recommendation engines, enhancing user engagement and satisfaction. Concurrently, attention is directed towards the user experience domain, analyzing interface design elements such as navigation schemes, search functionalities, and playlist customization options to understand how they influence user behavior and platform loyalty. Beyond the technological and experiential aspects, the survey extends its reach to the intricate business models that sustain these platforms, ranging from subscription-based models to ad-supported tiers and hybrid approaches, each posing unique challenges and opportunities in terms of revenue generation and user acquisition. Moreover, legal and copyright considerations form a pivotal part of the survey, encompassing the labyrinthine landscape of licensing agreements with record labels, publishers, and artists, alongside the ever-evolving legal frameworks governing digital rights management and copyright enforcement, which significantly impact platform operations and content availability. On a broader societal scale, the survey delves into the transformative effects of online music streaming on cultural consumption patterns, the democratization of music distribution, and its ramifications for industry stakeholders, including independent artists and niche genres. Furthermore, it explores the global dynamics of the streaming landscape, unveiling regional disparities in adoption rates, cultural preferences, and regulatory frameworks, thereby underscoring the need for localized strategies and content curation efforts to cater to diverse audiences worldwide. Finally, the survey navigates through the intricate interplay of market dynamics and competitive forces, unraveling strategic maneuvers by key players, market consolidation trends, and the disruptive potential of emerging technologies, all of which shape the future trajectory of the online music streaming ecosystem.

ENTITY-RELATIONSHIP DIAGRAM



ENTITIES AND THEIR ATTRIBUTES

Song	
Song_id	string pk
Name	string
Artist_id	string fk
likes	Int
Play_ctr	Int
picture	Link
storage_Link	Link

Artist	
Artist_id	string pk
Album_id	string fk
Category	string
Likes	Int
Play_ctr	Int
picture	Link

Playlist	
Playlist_id	string pk
Name	string
Category	string
Song_ID	string fk
Play_ctr	Int
Created_At	date
picture	Link

Users	
User_id	string pk
UserName	string
Name	string
DOB	date
gender	Bool
Profile_pic	Link
ArtistId	string

Album	
Album_id	string pk
Name	string
Artist_id	string fk
Play_ctr	Int
picture	Link

Login	
User_name	string pk
hashed_password	string

Create Queries

```
CREATE TABLE Artist (
    Artist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Category VARCHAR(50),
    Likes INT NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150)
);

CREATE TABLE Album (
    Album_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Artist_ID VARCHAR(50) NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);

CREATE TABLE Song (
    Song_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Artist_ID VARCHAR(50),
    Likes INT NOT NULL,
```

```
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    Storage_Link VARCHAR(150) NOT NULL,
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);

CREATE TABLE Playlist (
    Playlist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Category VARCHAR(50),
    Created_At DATE NOT NULL,
    Picture VARCHAR(150)
);

CREATE TABLE Customer (
    User_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    UserName VARCHAR(50) NOT NULL,
    Name VARCHAR(50) NOT NULL,
    DOB DATE NOT NULL,
    Gender VARCHAR(5) NOT NULL,
    Profile_Pic VARCHAR(150),
    Artist_ID VARCHAR(50) NOT NULL,
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);
```

```
CREATE TABLE Login (
    UserName VARCHAR(50) NOT NULL PRIMARY KEY,
    Hashed_Password VARCHAR(256) NOT NULL,
    User_ID VARCHAR(50) NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES Customer(User_ID)
);

CREATE TABLE Genre (
    Genre_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL
);

CREATE TABLE Subscription_Plan (
    Plan_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Price DECIMAL(10,2) NOT NULL
);

CREATE TABLE Subscription_History (
    Subscription_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    User_ID VARCHAR(50) NOT NULL,
    Plan_ID VARCHAR(50) NOT NULL,
    Start_Date DATE NOT NULL,
```

```
    End_Date DATE,
    FOREIGN KEY (User_ID) REFERENCES Customer(User_ID),
    FOREIGN KEY (Plan_ID) REFERENCES Subscription_Plan(Plan_ID)
);

CREATE TABLE Payment_Info (
    Payment_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    User_ID VARCHAR(50) NOT NULL,
    Payment_Method VARCHAR(50) NOT NULL,
    Card_Number VARCHAR(20) NOT NULL,
    Expiry_Date DATE NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES Customer(User_ID)
);

CREATE TABLE Playlist_Song (
    Playlist_ID VARCHAR(50) NOT NULL,
    Song_ID VARCHAR(50) NOT NULL,
    PRIMARY KEY (Playlist_ID, Song_ID),
    FOREIGN KEY (Playlist_ID) REFERENCES Playlist(Playlist_ID),
    FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)
);

CREATE TABLE Artist_Genre (
    Artist_ID VARCHAR(50) NOT NULL,
```

```
    Genre_ID VARCHAR(50) NOT NULL,
    PRIMARY KEY (Artist_ID, Genre_ID),
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID),
    FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
);

CREATE TABLE Album_Song (
    Album_ID VARCHAR(50) NOT NULL,
    Song_ID VARCHAR(50) NOT NULL,
    PRIMARY KEY (Album_ID, Song_ID),
    FOREIGN KEY (Album_ID) REFERENCES Album(Album_ID),
    FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)
);

CREATE TABLE Play_History (
    Play_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    User_ID VARCHAR(50) NOT NULL,
    Song_ID VARCHAR(50) NOT NULL,
    Play_Date TIMESTAMP NOT NULL,
    FOREIGN KEY (User_ID) REFERENCES Customer(User_ID),
    FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)
);

CREATE TABLE Advertisement (
```

```
    Ad_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Advertiser VARCHAR(50) NOT NULL,
    Duration INT NOT NULL,
    Ad_Link VARCHAR(150) NOT NULL,
    Start_Date DATE NOT NULL,
    End_Date DATE NOT NULL
);
```

Insert Queries

INSERT ALL

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR001', 'Taylor Swift', 'Pop', 100000, 5000000, 'https://some-image-link/taylor_swift.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR002', 'Ed Sheeran', 'Pop', 85000, 4200000, 'https://some-image-link/ed_sheeran.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR003', 'The Weeknd', 'R&B', 92000, 6000000, 'https://some-image-link/the_weeknd.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR004', 'Billie Eilish', 'Alternative', 75000, 3500000, 'https://some-image-link/billie_eilish.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR005', 'Drake', 'Hip-Hop', 120000, 7000000, 'https://some-image-link/drake.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR006', 'BTS', 'K-Pop', 150000, 8500000, 'https://some-image-link/bts.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR007', 'Ariana Grande', 'Pop', 98000, 5800000, 'https://some-image-link/ariana_grande.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR008', 'Post Malone', 'Hip-Hop', 80000, 4000000, 'https://some-image-link/post_malone.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR009', 'The Chainsmokers', 'EDM', 70000, 4500000, 'https://some-image-link/the_chainsmokers.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR010', 'Eminem', 'Hip-Hop', 110000, 6500000, 'https://some-image-link/eminem.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR011', 'Rihanna', 'R&B', 95000, 4800000, 'https://some-image-link/rihanna.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR012', 'Bruno Mars', 'Pop', 88000, 5500000, 'https://some-image-link/bruno_mars.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR013', 'Coldplay', 'Rock', 78000, 4200000, 'https://some-image-link/coldplay.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR014', 'Imagine Dragons', 'Rock', 72000, 3800000, 'https://some-image-link/imagine_dragons.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR015', 'Maroon 5', 'Pop', 82000, 4400000, 'https://some-image-link/maroon5.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR016', 'Adele', 'Pop', 90000, 5300000, 'https://some-image-link/adele.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR017', 'Justin Bieber', 'Pop', 105000, 6200000, 'https://some-image-link/justin_bieber.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR018', 'Bad Bunny', 'Reggaeton', 130000, 8000000, 'https://some-image-link/bad_bunny.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR019', 'Beyonce', 'R&B', 115000, 6800000, 'https://some-image-link/beyonce.jpg')

INTO Artist (Artist_ID, Name, Category, Likes, Play_ctr, Picture) VALUES ('AR020', 'Kendrick Lamar', 'Hip-Hop', 96000, 5500000, 'https://some-image-link/kendrick_lamar.jpg')

SELECT 1 FROM DUAL;

INSERT ALL

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL001', 'Lover', 'AR001', 3000000, 'https://some-image-link/lover.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL002', 'Reputation', 'AR001', 2500000, 'https://some-image-link/reputation.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL003', 'Divide', 'AR002', 2800000, 'https://some-image-link/divide.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL004', 'When We All Fall Asleep...', 'AR004', 1800000, 'https://some-image-link/wwafawdwg.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL005', 'Scorpion', 'AR005', 4500000, 'https://some-image-link/scorpion.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL006', 'Dynamite', 'AR006', 5600000, 'https://some-image-link/dynamite.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL007', 'Positions', 'AR007', 3200000, 'https://some-image-link/positions.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL008', 'Circles', 'AR008', 2100000, 'https://some-image-link/circles.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL009', 'Closer', 'AR009', 2900000, 'https://some-image-link/closer.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL010', 'Revival', 'AR010', 3300000, 'https://some-image-link/revival.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL011', 'Loud', 'AR011', 2400000, 'https://some-image-link/loud.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL012', '24K Magic', 'AR012', 3000000, 'https://some-image-link/24k_magic.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL013', 'AM', 'AR013', 2600000, 'https://some-image-link/am.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL014', 'Evolve', 'AR014', 2200000, 'https://some-image-link/evolve.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL015', 'Sugar', 'AR015', 2700000, 'https://some-image-link/sugar.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL016', '21', 'AR016', 2900000, 'https://some-image-link/21.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL017', 'Purpose', 'AR017', 3500000, 'https://some-image-link/purpose.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL018', 'El Dorado', 'AR018', 4000000, 'https://some-image-link/el_dorado.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL019', 'Queen', 'AR019', 3100000, 'https://some-image-link/queen.jpg')

INTO Album (Album_ID, Name, Artist_ID, Play_ctr, Picture) VALUES ('AL020', 'DAMN.', 'AR020', 3800000, 'https://some-image-link/damn.jpg')

SELECT 1 FROM DUAL;

INSERT ALL

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S001', 'Blank Space', 'AR001', 800000, 3500000, 'https://some-image-link/blank_space.jpg', 'https://some-audio-link/blank_space.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S002', 'Shape of You', 'AR002', 1200000, 5000000, 'https://some-image-link/shape_of_you.jpg', 'https://some-audio-link/shape_of_you.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S003', 'Bad Guy', 'AR004', 950000, 4200000, 'https://some-image-link/bad_guy.jpg', 'https://some-audio-link/bad_guy.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S004', 'Perfect', 'AR002', 1100000, 4800000, 'https://some-image-link/perfect.jpg', 'https://some-audio-link/perfect.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S005', 'God''s Plan', 'AR005', 1000000, 4500000, 'https://some-image-link/gods_plan.jpg', 'https://some-audio-link/gods_plan.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S006', 'Dynamite', 'AR006', 1500000, 6000000, 'https://some-image-link/dynamite.jpg', 'https://some-audio-link/dynamite.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S007', 'positions', 'AR007', 850000, 3600000, 'https://some-image-link/positions.jpg', 'https://some-audio-link/positions.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S008', 'Rockstar', 'AR008', 740000, 3200000, 'https://some-image-link/rockstar.jpg', 'https://some-audio-link/rockstar.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S009', 'Closer', 'AR009', 940000, 4000000, 'https://some-image-link/closer.jpg', 'https://some-audio-link/closer.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S010', 'Lose Yourself', 'AR010', 880000, 3900000, 'https://some-image-link/lose_yourself.jpg', 'https://some-audio-link/lose_yourself.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S011', 'Work', 'AR011', 770000, 3200000, 'https://some-image-link/work.jpg', 'https://some-audio-link/work.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S012', 'That''s What I Like', 'AR012', 900000, 4100000, 'https://some-image-link/thats_what_i_like.jpg', 'https://some-audio-link/thats_what_i_like.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S013', 'Do I Wanna Know?', 'AR013', 830000, 3800000, 'https://some-image-link/do_i_wanna_know.jpg', 'https://some-audio-link/do_i_wanna_know.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S014', 'Believer', 'AR014', 950000, 4400000, 'https://some-image-link/believer.jpg', 'https://some-audio-link/believer.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S015', 'V', 'AR015', 800000, 3600000, 'https://some-image-link/v.jpg', 'https://some-audio-link/v.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S016', 'Hello', 'AR016', 1100000, 4900000, 'https://some-image-link/hello.jpg', 'https://some-audio-link/hello.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S017', 'Sorry', 'AR017', 920000, 4200000, 'https://some-image-link/sorry.jpg', 'https://some-audio-link/sorry.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S018', 'Me Rehuso', 'AR018', 1200000, 5300000, 'https://some-image-link/me_rehuso.jpg', 'https://some-audio-link/me_rehuso.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S019', 'Bohemian Rhapsody', 'AR019', 850000, 3900000, 'https://some-image-link/bohemian_rhapsody.jpg', 'https://some-audio-link/bohemian_rhapsody.mp3')

INTO Song (Song_ID, Name, Artist_ID, Likes, Play_ctr, Picture, Storage_Link) VALUES ('S020', 'HUMBLE.', 'AR020', 890000, 4100000, 'https://some-image-link/humble.jpg', 'https://some-audio-link/humble.mp3')

SELECT 1 FROM DUAL;

INSERT ALL

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL001', 'Pop Hits 2024', 'Pop', TO_DATE('2024-03-27','YYYY-MM-DD'), 'https://some-image-link/pop_hits.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL002', 'My Workout Jams', 'Workout', TO_DATE('2024-02-10','YYYY-MM-DD'), 'https://some-image-link/workout_jams.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL003', 'Relaxing Vibes', 'Chill', TO_DATE('2024-03-12','YYYY-MM-DD'), 'https://some-image-link/relaxing_vibes.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL004', 'Tamil Classics', 'Regional', TO_DATE('2023-12-01','YYYY-MM-DD'), 'https://some-image-link/tamil_classics.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL005', 'Road Trip Essentials', 'Travel', TO_DATE('2024-03-18','YYYY-MM-DD'), 'https://some-image-link/road_trip.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL006', 'Hip-Hop Throwbacks', 'Hip-Hop', TO_DATE('2023-08-25','YYYY-MM-DD'), 'https://some-image-link/hiphop_throwbacks.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL007', 'Indie Favorites', 'Indie', TO_DATE('2024-01-15','YYYY-MM-DD'), 'https://some-image-link/indie_favorites.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL008', 'Romantic Evenings', 'Romance', TO_DATE('2024-02-14','YYYY-MM-DD'), 'https://some-image-link/romantic_evenings.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL009', 'EDM Beats', 'EDM', TO_DATE('2024-03-05','YYYY-MM-DD'), 'https://some-image-link/edm_beats.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL010', '90s Nostalgia', 'Pop', TO_DATE('2023-11-20','YYYY-MM-DD'), 'https://some-image-link/90s_nostalgia.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL011', 'Sunday Morning Coffee', 'Chill', TO_DATE('2024-03-10','YYYY-MM-DD'), 'https://some-image-link/sunday_coffee.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL012', 'Bollywood Hits', 'Regional', TO_DATE('2023-09-12','YYYY-MM-DD'), 'https://some-image-link/bollywood_hits.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL013', 'Focus Time', 'Instrumental', TO_DATE('2024-02-22','YYYY-MM-DD'), 'https://some-image-link/focus_time.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL014', 'Kids Singalong', 'Children', TO_DATE('2023-10-28','YYYY-MM-DD'), 'https://some-image-link/kids_singalong.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL015', 'Party Starters', 'Party', TO_DATE('2024-03-22','YYYY-MM-DD'), 'https://some-image-link/party_starters.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL016', 'Classical Favorites', 'Classical', TO_DATE('2023-07-14','YYYY-MM-DD'), 'https://some-image-link/classical_favorites.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL017', 'Rock Anthems', 'Rock', TO_DATE('2024-01-08','YYYY-MM-DD'), 'https://some-image-link/rock_anthems.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL018', 'Latin Heat', 'Latin', TO_DATE('2023-12-19','YYYY-MM-DD'), 'https://some-image-link/latin_heat.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL019', 'K-Pop Bops', 'K-Pop', TO_DATE('2024-03-01','YYYY-MM-DD'), 'https://some-image-link/kpop_bops.jpg')

INTO Playlist (Playlist_ID, Name, Category, Created_At, Picture) VALUES ('PL020', 'Country Roads', 'Country', TO_DATE('2024-02-18','YYYY-MM-DD'), 'https://some-image-link/country_roads.jpg')

SELECT 1 FROM DUAL;

INSERT ALL

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U001', 'musiclover24', 'Sarah Johnson', TO_DATE('1996-05-12','YYYY-MM-DD'), 'F', 'https://some-image-link/sarah_johnson.jpg', 'AR001')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U002', 'rockfanatic', 'Alex Miller', TO_DATE('1989-11-23','YYYY-MM-DD'), 'M', 'https://some-image-link/alex_miller.jpg', 'AR013')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U003', 'edm_enthusiast', 'Emily Davis', TO_DATE('1998-02-08','YYYY-MM-DD'), 'F', 'https://some-image-link/emily_davis.jpg', 'AR009')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U004', 'hiphophead', 'David Kim', TO_DATE('1992-08-15','YYYY-MM-DD'), 'M', 'https://some-image-link/david_kim.jpg', 'AR005')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U005', 'indielover', 'Mia Thompson', TO_DATE('1995-03-21','YYYY-MM-DD'), 'F', 'https://some-image-link/mia_thompson.jpg', 'AR014')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U006', 'kpopexplorer', 'Sofia Lee', TO_DATE('2000-10-04','YYYY-MM-DD'), 'F', 'https://some-image-link/sofia_lee.jpg', 'AR006')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U007', 'retrovibes', 'Noah Brown', TO_DATE('1985-06-26','YYYY-MM-DD'), 'M', 'https://some-image-link/noah_brown.jpg', 'AR019')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U008', 'latinbeats', 'Isabella Rodriguez', TO_DATE('1999-01-12','YYYY-MM-DD'), 'F', 'https://some-image-link/isabella_rodriguez.jpg', 'AR018')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U009', 'classicalconnoisseur', 'Jacob Wilson', TO_DATE('1980-09-18','YYYY-MM-DD'), 'M', 'https://some-image-link/jacob_wilson.jpg', 'AR016')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U010', 'countryfan', 'Olivia Patel', TO_DATE('1991-04-29','YYYY-MM-DD'), 'F', 'https://some-image-link/olivia_patel.jpg', 'AR008')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U011', 'tamil_musiclover', 'Priya Kumar', TO_DATE('1993-07-16','YYYY-MM-DD'), 'F', 'https://some-image-link/priya_kumar.jpg', 'AR015')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U012', 'bollywoodbuff', 'Rahul Sharma', TO_DATE('1997-12-01','YYYY-MM-DD'), 'M', 'https://some-image-link/rahul_sharma.jpg', 'AR017')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U013', 'jazzaddict', 'Emma Taylor', TO_DATE('1983-11-05','YYYY-MM-DD'), 'F', 'https://some-image-link/emma_taylor.jpg', 'AR012')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U014', 'focus_beats', 'Lucas Chen', TO_DATE('1994-08-29','YYYY-MM-DD'), 'M', 'https://some-image-link/lucas_chen.jpg', 'AR004')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U015', 'yogavibes', 'Chloe Nguyen', TO_DATE('1990-02-17','YYYY-MM-DD'), 'F', 'https://some-image-link/chloe_nguyen.jpg', 'AR002')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U016', 'workoutwarrior', 'Ethan Tremblay', TO_DATE('1995-09-03','YYYY-MM-DD'), 'M', 'https://some-image-link/ethan_tremblay.jpg', 'AR010')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U017', 'traveltunes', 'Ava Williams', TO_DATE('1996-01-30','YYYY-MM-DD'), 'F', 'https://some-image-link/ava_williams.jpg', 'AR007')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U018', 'kids_bop', 'Daniel Anderson', TO_DATE('2002-05-20','YYYY-MM-DD'), 'M', 'https://some-image-link/daniel_anderson.jpg', 'AR003')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U019', 'nostalgiabuff', 'Grace Patel', TO_DATE('1987-06-11','YYYY-MM-DD'), 'F', 'https://some-image-link/grace_patel.jpg', 'AR011')

INTO Customer (User_ID, UserName, Name, DOB, Gender, Profile_Pic, Artist_ID) VALUES ('U020', 'punjabifan', 'Harpreet Singh', TO_DATE('1990-10-18','YYYY-MM-DD'), 'M', 'https://some-image-link/harpreet_singh.jpg', 'AR020')

SELECT 1 FROM DUAL

```
INSERT ALL
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('musiclover24', '$2b$12$5ED3rE307Q54qjvEbrtVzuY3N3k5.wSKq7AOoV58Q8X.aO02zT32C', 'U001')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('rockfanatic', '$2b$12$f3sW2r.180l63hX1w98Psu091mN8kG93bFtuw4Yc02H83F.WqvX1a', 'U002')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('edm_enthusiast', '$2b$12$6X18b09v908v73hL6vGusehUqV892N1h810Wws123S65tZ87a921O', 'U003')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('hiphophead', '$2b$12$H081b6f20t810hH630gFvu1jJ1l780sT756h123kM56v1iO917X8o', 'U004')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('indielover', '$2b$12$gN56o9980a12kM098tT123uKq65o321lN78078vG2fW1a4T.3l55O', 'U005')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('kpopexplorer', '$2b$12$Y2k0vV7u12lK65sJ6u7vO528l09kP312hH08hJ76lL820zM981k76', 'U006')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('retrovibes', '$2b$12$f608pW1u6pE08hK65pR09pO43p09w6eU9sQ45i0983k4fZ8y2k82a', 'U007')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('latinbeats', '$2b$12$g1820gF23lE23hU23rT0723tU23tU45nG65oU65t5jY28jH201k.l', 'U008')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('classicalconnoisseur', '$2b$12$7sE612u871uB1bS612bU43bU34nR54uH54vH5nU54yA34zY12812u', 'U009')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('countryfan', '$2b$12$b0w912hQ64wR34hF73tH2734hL56bL32bS5jJ4jH82jS1jN352z9o', 'U010')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('tamil_musiclover', '$2b$12$gT43bG54bT23hU12bS12sL43vT4vF5bS34uU54hN80iG45f252910', 'U011')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('bollywoodbuff', '$2b$12$1jN32jH1jL2j3vH80yV45bT63sS51kQ1kU2bW45uE12jS890s19qF', 'U012')
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('jazzaddict', '$2b$12$sT62hK43sN12hK12nJ8qW08eW7qT76tS6tQ57vG5vN54vA65t42.1', 'U013')
```

```
  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('focus_beats', '$2b$12$lA9s76vG12eT09wS65wQ54wF23qD32sF67uG7gO623oH8a92z5121', 'U014')

  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('yogavibes', '$2b$12$vG54bT3rQ54tF6sQ45wQ43wD23rT2vG6vG5nB6sN6rT6bG2b12s1z', 'U015')

  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('workoutwarrior', '$2b$12$gV43bH3qF5vV64wT7rS3rS12qR3qR53qS2qH3qG5uG56qG6q2z.12', 'U016')

  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('traveltunes', '$2b$12$2zX3q7zX5nB6sN4rR5vE43rT3qQ53qS46wR56sG45sF612vA231u.', 'U017')

  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('kids_bop', '$2b$12$bN76sB23vG4uG43tU4qT3qS3qR1rS43sN7bT47uT7rS6bV9p12o91', 'U018')

  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('nostalgiabuff', '$2b$12$1hL3jK1jS1lK2jN4jK5lJ43lT3uV08uV7sU77qS7sN9sG412a521u', 'U019')

  INTO Login (UserName, Hashed_Password, User_ID) VALUES ('punjabifan', '$2b$12$qB6sN7rQ6wR6rT2qS2qS34sF46tF64wR2qS23sE53sG34zX21a1a', 'U020')

SELECT 1 FROM DUAL;



INSERT ALL

  INTO Genre (Genre_ID, Name) VALUES ('G001', 'Pop')

  INTO Genre (Genre_ID, Name) VALUES ('G002', 'Rock')

  INTO Genre (Genre_ID, Name) VALUES ('G003', 'Hip-Hop')

  INTO Genre (Genre_ID, Name) VALUES ('G004', 'R&B')

  INTO Genre (Genre_ID, Name) VALUES ('G005', 'EDM')

  INTO Genre (Genre_ID, Name) VALUES ('G006', 'Country')

  INTO Genre (Genre_ID, Name) VALUES ('G007', 'Indie')
```

```
  INTO Genre (Genre_ID, Name) VALUES ('G008', 'K-Pop')
  INTO Genre (Genre_ID, Name) VALUES ('G009', 'Latin')
  INTO Genre (Genre_ID, Name) VALUES ('G010', 'Classical')
  INTO Genre (Genre_ID, Name) VALUES ('G011', 'Jazz')
  INTO Genre (Genre_ID, Name) VALUES ('G012', 'Reggaeton')
  INTO Genre (Genre_ID, Name) VALUES ('G013', 'Bollywood')
  INTO Genre (Genre_ID, Name) VALUES ('G014', 'Tamil')
  INTO Genre (Genre_ID, Name) VALUES ('G015', 'Punjabi')
  INTO Genre (Genre_ID, Name) VALUES ('G016', 'Instrumental')
  INTO Genre (Genre_ID, Name) VALUES ('G017', 'Children')
  INTO Genre (Genre_ID, Name) VALUES ('G018', 'Workout')
  INTO Genre (Genre_ID, Name) VALUES ('G019', 'Chill')
  INTO Genre (Genre_ID, Name) VALUES ('G020', 'Regional')
SELECT 1 FROM DUAL;



INSERT ALL
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P001', 'Free', 0.00)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P002', 'Basic', 4.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P003', 'Standard', 9.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P004', 'Premium', 14.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P005', 'Family', 19.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P006', 'Student', 6.99)
```

```
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P007', 'Basic Annual', 49.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P008', 'Standard Annual', 99.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P009', 'Premium Annual', 149.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P010', 'Regional Basic', 3.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P011', 'Regional Standard', 7.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P012', 'Regional Premium', 12.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P013', 'Trial 30 Days', 0.00)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P014', 'Trial 7 Days', 0.00)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P015', 'Business Basic', 24.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P016', 'Business Standard', 49.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P017', 'Business Premium', 99.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P018', 'Artist Basic', 12.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P019', 'Artist Premium', 24.99)
  INTO Subscription_Plan (Plan_ID, Name, Price) VALUES ('P020', 'Lifetime', 199.99)
SELECT 1 FROM DUAL;



INSERT ALL
```

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH001', 'U001', 'P002', TO_DATE('2024-02-15','YYYY-MM-DD'), TO_DATE('2024-03-15','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH002', 'U003', 'P003', TO_DATE('2024-03-20','YYYY-MM-DD'), NULL)

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH003', 'U005', 'P001', TO_DATE('2023-12-04','YYYY-MM-DD'), TO_DATE('2024-01-04','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH004', 'U005', 'P004', TO_DATE('2024-01-05','YYYY-MM-DD'), TO_DATE('2024-02-05','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH005', 'U007', 'P001', TO_DATE('2023-11-28','YYYY-MM-DD'), TO_DATE('2023-12-28','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH006', 'U007', 'P013', TO_DATE('2023-12-29','YYYY-MM-DD'), TO_DATE('2024-01-28','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH007', 'U002', 'P006', TO_DATE('2024-03-12','YYYY-MM-DD'), NULL)

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH008', 'U009', 'P008', TO_DATE('2023-05-17','YYYY-MM-DD'), TO_DATE('2024-05-16','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH009', 'U011', 'P010', TO_DATE('2024-01-22','YYYY-MM-DD'), TO_DATE('2024-02-22','YYYY-MM-DD'))

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH010', 'U011', 'P011', TO_DATE('2024-02-23','YYYY-MM-DD'), NULL)

INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH011', 'U015', 'P001', TO_DATE('2023-08-04','YYYY-MM-DD'), TO_DATE('2023-09-04','YYYY-MM-DD'))

```
  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH012', 'U015', 'P001', TO_DATE('2023-10-15','YYYY-MM-DD'), TO_DATE('2023-11-15','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH013', 'U015', 'P003', TO_DATE('2023-11-16','YYYY-MM-DD'), TO_DATE('2024-01-16','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH014', 'U018', 'P001', TO_DATE('2023-06-26','YYYY-MM-DD'), TO_DATE('2024-06-25','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH015', 'U004', 'P015', TO_DATE('2023-09-11','YYYY-MM-DD'), TO_DATE('2024-03-10','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH016', 'U006', 'P005', TO_DATE('2024-02-29','YYYY-MM-DD'), NULL)

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH017', 'U017', 'P003', TO_DATE('2023-12-01','YYYY-MM-DD'), TO_DATE('2024-01-01','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH018', 'U013', 'P001', TO_DATE('2023-10-05','YYYY-MM-DD'), TO_DATE('2023-11-05','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH019', 'U013', 'P003', TO_DATE('2023-11-06','YYYY-MM-DD'), TO_DATE('2023-12-06','YYYY-MM-DD'))

  INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date, End_Date) VALUES ('SH020', 'U020', 'P012', TO_DATE('2024-03-05','YYYY-MM-DD'), NULL)

SELECT 1 FROM DUAL;
```

```
INSERT ALL
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL001', 'S001')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL001', 'S002')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL001', 'S004')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL001', 'S007')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL002', 'S005')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL002', 'S008')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL002', 'S009')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL002', 'S012')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL003', 'S003')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL003', 'S006')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL003', 'S016')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL003', 'S019')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL004', 'S015')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL004', 'S018')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL004', 'S020')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL005', 'S002')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL005', 'S005')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL005', 'S010')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL005', 'S013')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL006', 'S008')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL006', 'S010')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL006', 'S020')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL007', 'S003')
```

```
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL007', 'S013')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL007', 'S014')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL008', 'S004')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL008', 'S007')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL008', 'S016')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL009', 'S006')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL009', 'S009')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL009', 'S012')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL010', 'S009')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL010', 'S011')
  INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('PL010', 'S019')
SELECT 1 FROM DUAL;



INSERT ALL
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR001', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR002', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR002', 'G019')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR003', 'G004')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR004', 'G019')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR004', 'G016')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR005', 'G003')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR006', 'G008')
```

```
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR007', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR008', 'G003')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR009', 'G005')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR010', 'G003')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR011', 'G004')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR011', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR012', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR013', 'G002')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR014', 'G002')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR015', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR016', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR017', 'G001')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR018', 'G009')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR019', 'G004')
  INTO Artist_Genre (Artist_ID, Genre_ID) VALUES ('AR020', 'G003')
SELECT 1 FROM DUAL;



INSERT ALL
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL001', 'S001')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL001', 'S002')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL001', 'S004')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL002', 'S002')
```

```
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL002', 'S004')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL003', 'S003')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL004', 'S005')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL005', 'S006')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL006', 'S007')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL007', 'S008')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL008', 'S009')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL009', 'S010')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL010', 'S011')
  INTO Album_Song (Album_ID, Song_ID) VALUES ('AL011', 'S012')
SELECT 1 FROM DUAL;



INSERT ALL
  INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH001', 'U001',
'S002', TIMESTAMP '2024-03-26 10:32:00')
  INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH002', 'U003',
'S006', TIMESTAMP '2024-03-27 08:15:00')
  INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH003', 'U005',
'S014', TIMESTAMP '2024-03-25 19:45:00')
  INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH004', 'U002',
'S008', TIMESTAMP '2024-03-26 12:02:00')
  INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH005', 'U009',
'S019', TIMESTAMP '2024-03-26 07:30:00')
```

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH006', 'U009', 'S016', TIMESTAMP '2024-03-27 07:35:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH007', 'U004', 'S005', TIMESTAMP '2024-03-25 21:05:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH008', 'U001', 'S001', TIMESTAMP '2024-03-27 09:20:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH009', 'U006', 'S006', TIMESTAMP '2024-03-26 14:38:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH010', 'U008', 'S018', TIMESTAMP '2024-03-25 18:00:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH011', 'U012', 'S017', TIMESTAMP '2024-03-27 11:12:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH012', 'U015', 'S003', TIMESTAMP '2024-03-26 15:45:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH013', 'U015', 'S009', TIMESTAMP '2024-03-26 15:56:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH014', 'U010', 'S008', TIMESTAMP '2024-03-25 09:05:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH015', 'U013', 'S013', TIMESTAMP '2024-03-27 17:30:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH016', 'U014', 'S003', TIMESTAMP '2024-03-26 20:12:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH017', 'U017', 'S007', TIMESTAMP '2024-03-25 12:45:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH018', 'U007', 'S011', TIMESTAMP '2024-03-27 16:05:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH019', 'U019', 'S011', TIMESTAMP '2024-03-26 11:11:00')

INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PH020', 'U020', 'S020', TIMESTAMP '2024-03-25 15:59:00')

SELECT 1 FROM DUAL;

INSERT ALL

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD001', 'Spotify', 30, 'https://spotify-ad.com/latest-promo', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD002', 'Apple Music', 30, 'https://apple-music-ad.com/free-trial', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD003', 'Local Gym', 15, 'https://local-gym.com/membership-deals', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD004', 'Headspace', 20, 'https://headspace-ad.com/mindfulness', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD005', 'Masterclass', 30, 'https://masterclass-ad.com/learn-online', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD006', 'Nike', 15, 'https://nike-ad.com/new-releases', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD007', 'Amazon Prime', 30, 'https://amazon-prime-ad.com/join', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD008', 'Local Coffee Shop', 15, 'https://local-coffee.com/new-blends', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD009', 'Grammarly', 20, 'https://grammarly-ad.com/write-better', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD010', 'Audible', 30, 'https://audible-ad.com/free-audiobook', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD011', 'Meal Delivery Service', 30, 'https://meal-delivery-ad.com/discount', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD012', 'Online Course Platform', 20, 'https://online-courses-ad.com/skills', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD013', 'Travel Agency', 30, 'https://travel-agency-ad.com/packages', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD014', 'VPN Service', 20, 'https://vpn-ad.com/privacy', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD015', 'Mobile Game', 15, 'https://mobile-game-ad.com/download', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD016', 'Charity Organization', 30, 'https://charity-ad.com/donate', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD017', 'Online Store', 20, 'https://online-store-ad.com/sale', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD018', 'Language Learning App', 30, 'https://language-app-ad.com/fluent', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD019', 'Food Delivery App', 15, 'https://food-delivery-ad.com/first-free', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('AD020', 'Tech Gadget', 20, 'https://tech-gadget-ad.com/latest', TO_DATE('1989-12-09','YYYY-MM-DD'), TO_DATE('1989-12-09','YYYY-MM-DD'))

SELECT 1 FROM DUAL;

INSERT ALL

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY001', 'U001', 'Visa', 'XXXX-XXXX-XXXX-1234', TO_DATE('2025-12-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY002', 'U002', 'Mastercard', 'XXXX-XXXX-XXXX-5678', TO_DATE('2026-05-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY003', 'U003', 'PayPal', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2024-09-30','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY004', 'U005', 'Visa', 'XXXX-XXXX-XXXX-9012', TO_DATE('2027-01-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY005', 'U006', 'American Express', 'XXXX-XXXX-XXXX-3456', TO_DATE('2024-11-30','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY006', 'U007', 'PayPal', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2025-06-30','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY007', 'U009', 'Mastercard', 'XXXX-XXXX-XXXX-7890', TO_DATE('2026-08-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY008', 'U010', 'Visa', 'XXXX-XXXX-XXXX-2468', TO_DATE('2025-03-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY009', 'U012', 'PayPal', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2024-04-30','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY010', 'U014', 'Mastercard', 'XXXX-XXXX-XXXX-1357', TO_DATE('2027-10-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY011', 'U004', 'Visa', 'XXXX-XXXX-XXXX-9876', TO_DATE('2026-12-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY012', 'U008', 'Apple Pay', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2025-09-30','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY013', 'U016', 'Mastercard', 'XXXX-XXXX-XXXX-5432', TO_DATE('2024-07-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY014', 'U017', 'PayPal', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2026-02-28','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY015', 'U018', 'Visa', 'XXXX-XXXX-XXXX-8642', TO_DATE('2027-05-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY016', 'U006', 'Google Pay', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2025-11-30','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY017', 'U013', 'Mastercard', 'XXXX-XXXX-XXXX-6543', TO_DATE('2024-02-28','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY018', 'U020', 'Visa', 'XXXX-XXXX-XXXX-0000', TO_DATE('2026-08-31','YYYY-MM-DD'))

INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY019', 'U005', 'PayPal', 'XXXX-XXXX-XXXX-XXXX', TO_DATE('2025-03-31','YYYY-MM-DD'))

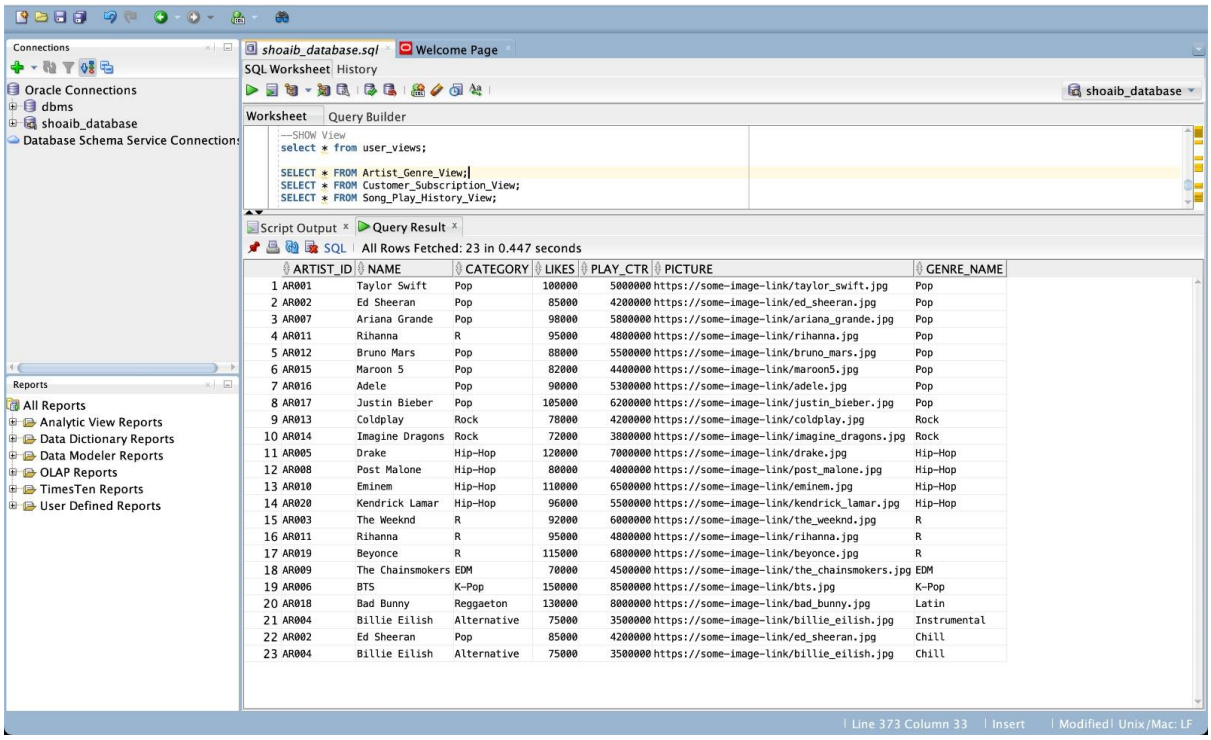
INTO Payment_Info (Payment_ID, User_ID, Payment_Method, Card_Number, Expiry_Date) VALUES ('PAY020', 'U015', 'Mastercard', 'XXXX-XXXX-XXXX-4321', TO_DATE('2027-06-30','YYYY-MM-DD')

SELECT 1 FROM DUAL;

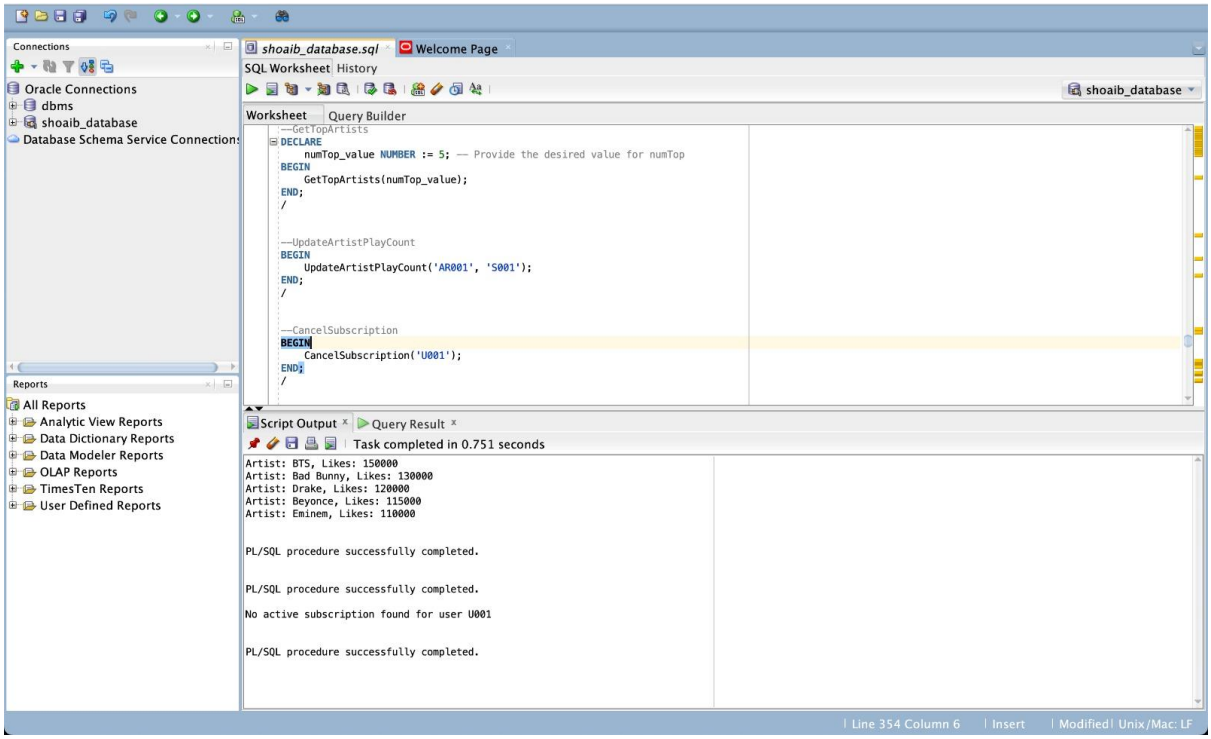
VIEWS

CREATE VIEW Artist_Genre_View AS

```
SELECT A.*, G.Name AS Genre_Name
FROM Artist A
JOIN Artist_Genre AG ON A.Artist_ID = AG.Artist_ID
JOIN Genre G ON AG.Genre_ID = G.Genre_ID;


CREATE VIEW Customer_Subscription_View AS
SELECT C.*, SH.Start_Date AS Subscription_Start_Date, SH.End_Date AS Subscription_End_Date,
    SP.Name AS Subscription_Plan_Name, SP.Price AS Subscription_Plan_Price
FROM Customer C
LEFT JOIN Subscription_History SH ON C.User_ID = SH.User_ID
LEFT JOIN Subscription_Plan SP ON SH.Plan_ID = SP.Plan_ID;


CREATE VIEW Song_Play_History_View AS
SELECT PH.*, C.UserName AS Customer_Username, C.Name AS Customer_Name
FROM Play_History PH
JOIN Customer C ON PH.User_ID = C.User_ID;
```



	ARTIST_ID	NAME	CATEGORY	LIKES	PLAY_CTR	PICTURE	GENRE_NAME
1	AR001	Taylor Swift	Pop	100000	5000000	https://some-image-link/taylor_swift.jpg	Pop
2	AR002	Ed Sheeran	Pop	85000	4200000	https://some-image-link/ed_sheeran.jpg	Pop
3	AR007	Ariana Grande	Pop	98000	5800000	https://some-image-link/ariana_grande.jpg	Pop
4	AR011	Rihanna	R	95000	4800000	https://some-image-link/rihanna.jpg	Pop
5	AR012	Bruno Mars	Pop	88000	5500000	https://some-image-link/bruno_mars.jpg	Pop
6	AR015	Maroon 5	Pop	82000	4400000	https://some-image-link/maroon5.jpg	Pop
7	AR016	Adele	Pop	90000	5300000	https://some-image-link/adele.jpg	Pop
8	AR017	Justin Bieber	Pop	105000	6200000	https://some-image-link/justin_bieber.jpg	Pop
9	AR013	Coldplay	Rock	78000	4200000	https://some-image-link/coldplay.jpg	Rock
10	AR014	Imagine Dragons	Rock	72000	3800000	https://some-image-link/imagine_dragons.jpg	Rock
11	AR005	Drake	Hip-Hop	120000	7000000	https://some-image-link/drake.jpg	Hip-Hop
12	AR008	Post Malone	Hip-Hop	80000	4000000	https://some-image-link/post_malone.jpg	Hip-Hop
13	AR010	Eminem	Hip-Hop	110000	6500000	https://some-image-link/eminem.jpg	Hip-Hop
14	AR020	Kendrick Lamar	Hip-Hop	96000	5500000	https://some-image-link/kendrick_lamar.jpg	Hip-Hop
15	AR003	The Weeknd	R	92000	6000000	https://some-image-link/the_weeknd.jpg	R
16	AR011	Rihanna	R	95000	4800000	https://some-image-link/rihanna.jpg	R
17	AR019	Beyonce	R	115000	6800000	https://some-image-link/beyonce.jpg	R
18	AR009	The Chainsmokers	EDM	70000	4500000	https://some-image-link/the_chainsmokers.jpg	EDM
19	AR006	BTS	K-Pop	150000	8500000	https://some-image-link/bts.jpg	K-Pop
20	AR018	Bad Bunny	Reggaeton	130000	8000000	https://some-image-link/bad_bunny.jpg	Latin
21	AR004	Billie Eilish	Alternative	75000	3500000	https://some-image-link/billie_eilish.jpg	Instrumental
22	AR002	Ed Sheeran	Pop	85000	4200000	https://some-image-link/ed_sheeran.jpg	Chill
23	AR004	Billie Eilish	Alternative	75000	3500000	https://some-image-link/billie_eilish.jpg	Chill



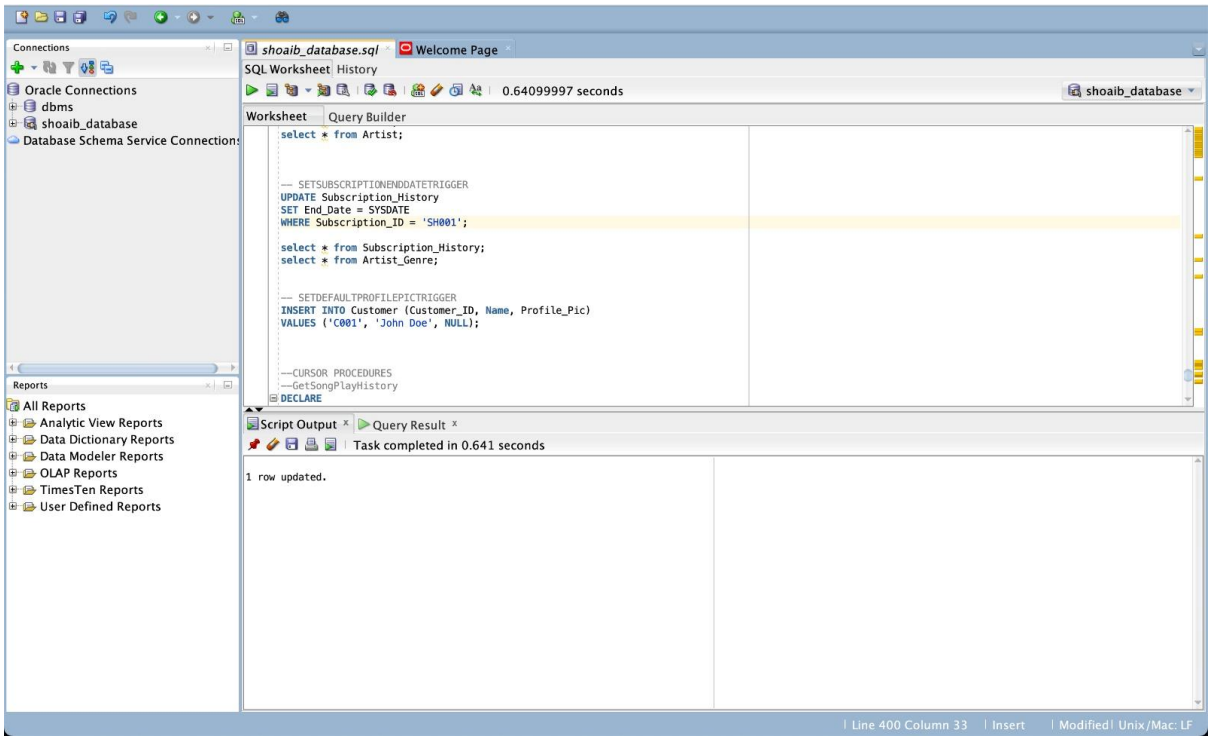
PROCEDURES

```
--Procedure 1 -> UpdateArtistPlayCount
DROP PROCEDURE UpdateArtistPlayCount;

CREATE OR REPLACE PROCEDURE UpdateArtistPlayCount(
    artistID IN VARCHAR2,
    songID IN VARCHAR2
)
IS
    albumID VARCHAR2(50);
BEGIN
    -- Get the album ID of the song
    SELECT Album_ID INTO albumID FROM Album_Song WHERE Song_ID = songID;

    -- Update the play count of the artist
    UPDATE Artist
    SET Play_ctr = Play_ctr + 1
    WHERE Artist_ID = artistID;
```

```
    -- Update the play count of the album
    UPDATE Album
    SET Play_ctr = Play_ctr + 1
    WHERE Album_ID = albumID;
END;
/
CREATE OR REPLACE PROCEDURE CancelSubscription(
    userID IN VARCHAR2
)
IS
    subscriptionID VARCHAR2(50);
BEGIN
    -- Get the subscription ID of the user
    SELECT Subscription_ID INTO subscriptionID
    FROM Subscription_History
    WHERE User_ID = userID
    AND End_Date IS NULL;

    -- Update the end date of the subscription history
    UPDATE Subscription_History
    SET End_Date = SYSDATE
    WHERE User_ID = userID
    AND Subscription_ID = subscriptionID;
END;
```

```
CREATE OR REPLACE PROCEDURE GetTopArtists(
    numTop IN NUMBER
)
IS
    CURSOR c_top_artists IS
        SELECT *
        FROM Artist
        ORDER BY Likes DESC
        FETCH FIRST numTop ROWS ONLY;
    v_artist Artist%ROWTYPE;
BEGIN
    OPEN c_top_artists;
    LOOP
        FETCH c_top_artists INTO v_artist;
        EXIT WHEN c_top_artists%NOTFOUND;
        -- Process the top artist record here (e.g., display, store, etc.)
        DBMS_OUTPUT.PUT_LINE('Artist: ' || v_artist.Name || ', Likes: ' || v_artist.Likes);
    END LOOP;
    CLOSE c_top_artists;
END;
/
--List all Procedures
SELECT
```

```
    *
FROM
    user_procedures
WHERE
    object_type = 'PROCEDURE';
```



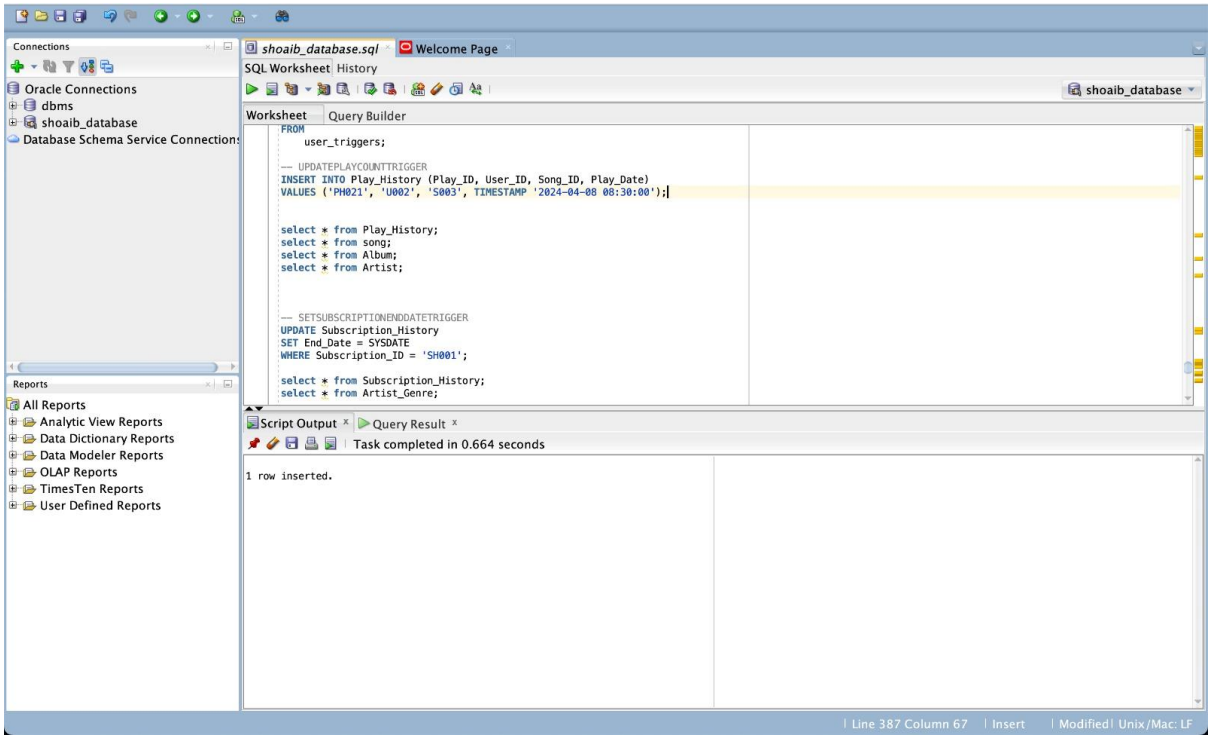
CURSORS

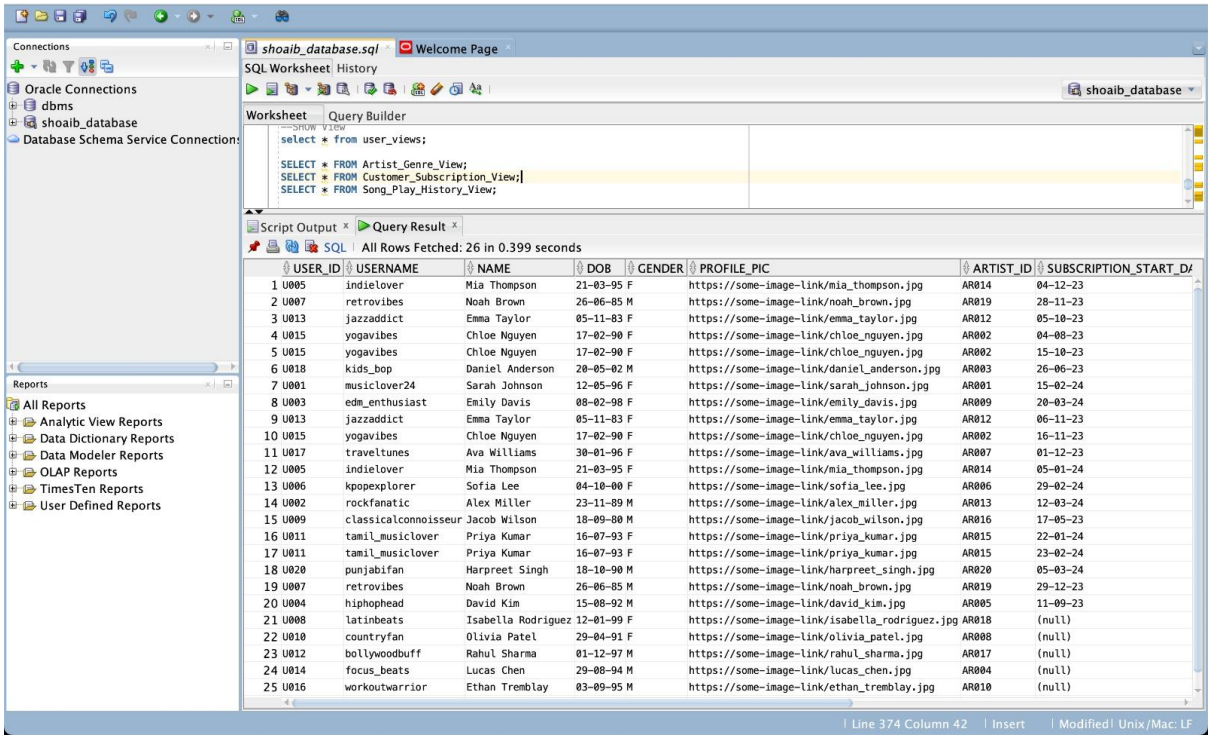
```
CREATE OR REPLACE PROCEDURE GetPlaylistInfo(
    playlistID IN VARCHAR2
)
```

```
IS
    CURSOR c_playlist_info IS
        SELECT P.*, S.Name AS Song_Name
        FROM Playlist P
        JOIN Playlist_Song PS ON P.Playlist_ID = PS.Playlist_ID
        JOIN Song S ON PS.Song_ID = S.Song_ID
        WHERE P.Playlist_ID = playlistID;
    v_playlist_info c_playlist_info%ROWTYPE;
BEGIN
    OPEN c_playlist_info;
    FETCH c_playlist_info INTO v_playlist_info;
    IF c_playlist_info%FOUND THEN
        -- Process the playlist information here (e.g., display, store, etc.)
        DBMS_OUTPUT.PUT_LINE('Playlist Name: ' || v_playlist_info.Name);
        DBMS_OUTPUT.PUT_LINE('Category: ' || v_playlist_info.Category);
        DBMS_OUTPUT.PUT_LINE('Created At: ' || TO_CHAR(v_playlist_info.Created_At,
'YYYY-MM-DD'));
        DBMS_OUTPUT.PUT_LINE('Songs:');
        WHILE c_playlist_info%FOUND LOOP
            DBMS_OUTPUT.PUT_LINE(' - ' || v_playlist_info.Song_Name);
            FETCH c_playlist_info INTO v_playlist_info;
        END LOOP;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Playlist not found.');
    END IF;
```

```
    CLOSE c_playlist_info;
END;
CREATE OR REPLACE PROCEDURE CalculateAveragePlayCount
IS
    total_play_count NUMBER := 0;
    total_artists NUMBER := 0;
    v_artist_play_count NUMBER;
    CURSOR c_artist_play_count IS
        SELECT Play_ctr
        FROM Artist;
BEGIN
    OPEN c_artist_play_count;
    LOOP
        FETCH c_artist_play_count INTO v_artist_play_count;
        EXIT WHEN c_artist_play_count%NOTFOUND;
        total_play_count := total_play_count + v_artist_play_count;
        total_artists := total_artists + 1;
    END LOOP;
    CLOSE c_artist_play_count;

    IF total_artists > 0 THEN
        DBMS_OUTPUT.PUT_LINE('Average play count of all artists: ' || total_play_count /
total_artists);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No artists found.');
```

```
    END IF;
END;



CREATE OR REPLACE PROCEDURE GetSongPlayHistory(
    songID IN VARCHAR2
)
IS
    CURSOR c_play_history IS
        SELECT *
        FROM Play_History
        WHERE Song_ID = songID;
    v_play_history Play_History%ROWTYPE;
BEGIN
    OPEN c_play_history;
    LOOP
        FETCH c_play_history INTO v_play_history;
        EXIT WHEN c_play_history%NOTFOUND;
        -- Process the play history record here (e.g., display, store, etc.)
        DBMS_OUTPUT.PUT_LINE('User ' || v_play_history.User_ID || ' played the song on ' ||
TO_CHAR(v_play_history.Play_Date, 'YYYY-MM-DD HH24:MI:SS'));
    END LOOP;
    CLOSE c_play_history;
END;
```



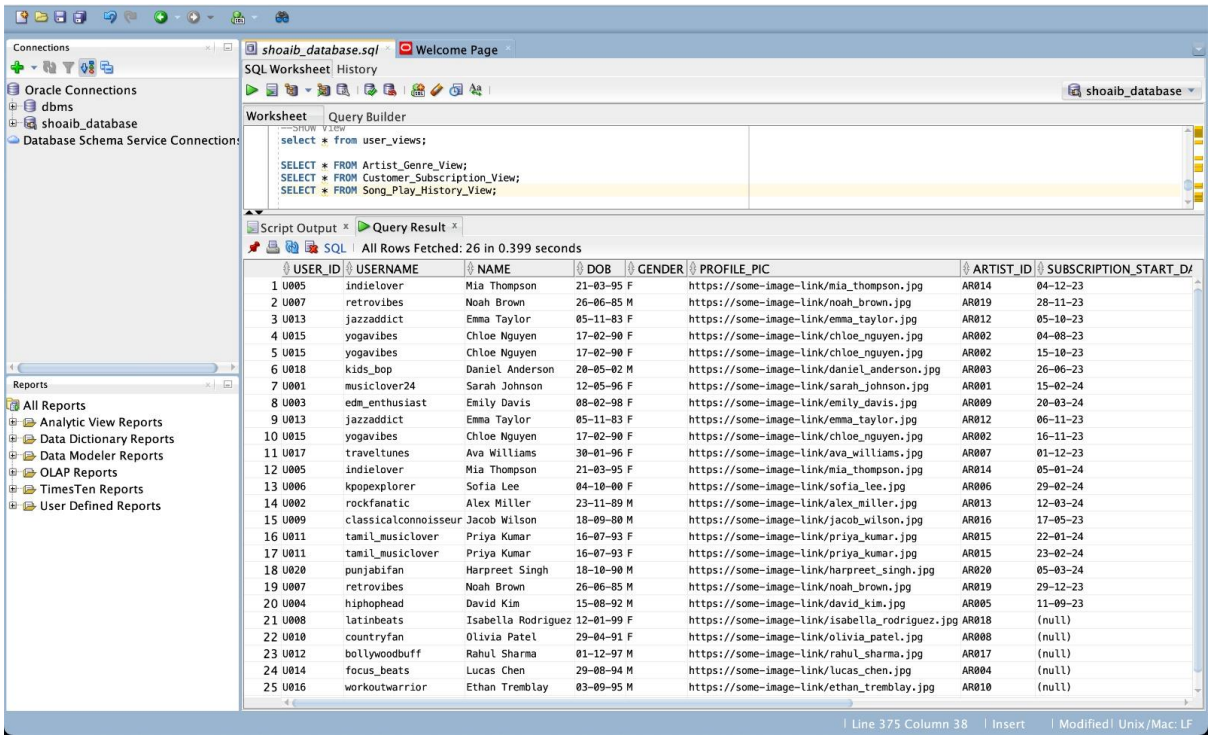


	USER_ID	USERNAME	NAME	DOB	GENDER	PROFILE_PIC	ARTIST_ID	SUBSCRIPTION_START_DA
1	U005	indielover	Mia Thompson	21-03-95	F	https://some-image-link/mia_thompson.jpg	AR014	04-12-23
2	U007	retrovibes	Noah Brown	26-06-85	M	https://some-image-link/noah_brown.jpg	AR019	28-11-23
3	U013	jazzaddict	Emma Taylor	05-11-83	F	https://some-image-link/emma_taylor.jpg	AR012	05-10-23
4	U015	yogavibes	Chloe Nguyen	17-02-90	F	https://some-image-link/chloe_nguyen.jpg	AR002	04-08-23
5	U015	yogavibes	Chloe Nguyen	17-02-90	F	https://some-image-link/chloe_nguyen.jpg	AR002	15-10-23
6	U018	kids_bop	Daniel Anderson	20-05-02	M	https://some-image-link/daniel_anderson.jpg	AR003	26-06-23
7	U001	musiclover24	Sarah Johnson	12-05-96	F	https://some-image-link/sarah_johnson.jpg	AR001	15-02-24
8	U003	edm_enthusiast	Emily Davis	08-02-98	F	https://some-image-link/emily_davis.jpg	AR009	20-03-24
9	U013	jazzaddict	Emma Taylor	05-11-83	F	https://some-image-link/emma_taylor.jpg	AR012	06-11-23
10	U015	yogavibes	Chloe Nguyen	17-02-90	F	https://some-image-link/chloe_nguyen.jpg	AR002	16-11-23
11	U017	traveltunes	Ava Williams	30-01-96	F	https://some-image-link/ava_williams.jpg	AR007	01-12-23
12	U005	indielover	Mia Thompson	21-03-95	F	https://some-image-link/mia_thompson.jpg	AR014	05-01-24
13	U006	kpopexplorer	Sofia Lee	04-10-00	F	https://some-image-link/sofia_lee.jpg	AR006	29-02-24
14	U002	rockfanatic	Alex Miller	23-11-89	M	https://some-image-link/alex_miller.jpg	AR013	12-03-24
15	U009	classicalconnoisseur	Jacob Wilson	18-09-80	M	https://some-image-link/jacob_wilson.jpg	AR016	17-05-23
16	U011	tamil_musiclover	Priya Kumar	16-07-93	F	https://some-image-link/priya_kumar.jpg	AR015	22-01-24
17	U011	tamil_musiclover	Priya Kumar	16-07-93	F	https://some-image-link/priya_kumar.jpg	AR015	23-02-24
18	U020	punjabifan	Harpreet Singh	18-10-90	M	https://some-image-link/harpreet_singh.jpg	AR020	05-03-24
19	U007	retrovibes	Noah Brown	26-06-85	M	https://some-image-link/noah_brown.jpg	AR019	29-12-23
20	U004	hiphophead	David Kim	15-08-92	M	https://some-image-link/david_kim.jpg	AR005	11-09-23
21	U008	latinbeats	Isabella Rodriguez	12-01-99	F	https://some-image-link/isabella_rodriguez.jpg	AR018	(null)
22	U010	countryfan	Olivia Patel	29-04-91	F	https://some-image-link/olivia_patel.jpg	AR008	(null)
23	U012	bollywoodbuff	Rahul Sharma	01-12-97	M	https://some-image-link/rahul_sharma.jpg	AR017	(null)
24	U014	focus_beats	Lucas Chen	29-08-94	M	https://some-image-link/lucas_chen.jpg	AR004	(null)
25	U016	workoutwarrior	Ethan Tremblay	03-09-95	M	https://some-image-link/ethan_tremblay.jpg	AR010	(null)

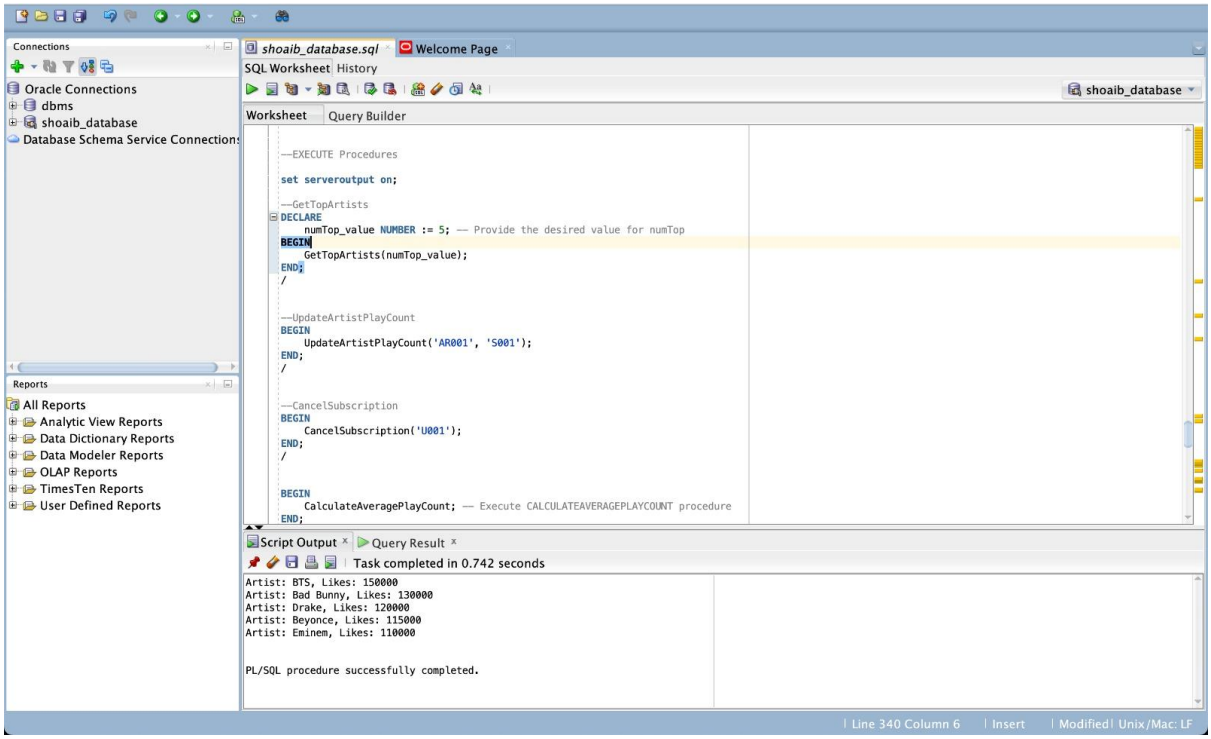
TRIGGERS

```
CREATE OR REPLACE TRIGGER SetDefaultProfilePicTrigger
BEFORE INSERT ON Customer
FOR EACH ROW
BEGIN
   IF :NEW.Profile_Pic IS NULL THEN
      :NEW.Profile_Pic := 'https://example.com/default_profile_pic.jpg';
   END IF;
END;
CREATE OR REPLACE TRIGGER SetSubscriptionEndDateTrigger
BEFORE UPDATE OF End_Date ON Subscription_History
FOR EACH ROW
BEGIN
   IF :NEW.End_Date IS NOT NULL THEN
      :NEW.End_Date := SYSDATE;
   END IF;
END;
CREATE OR REPLACE TRIGGER UpdatePlayCountTrigger
AFTER INSERT ON Play_History
FOR EACH ROW
DECLARE
```

```
    v_albumID VARCHAR(50);
BEGIN
    -- Get the album ID of the played song
    SELECT Album_ID INTO v_albumID
    FROM Album_Song
    WHERE Song_ID = :NEW.Song_ID;

    -- Update the play count of the artist
    UPDATE Artist
    SET Play_ctr = Play_ctr + 1
    WHERE Artist_ID = (SELECT Artist_ID FROM Song WHERE Song_ID =
:NEW.Song_ID);

    -- Update the play count of the album
    UPDATE Album
    SET Play_ctr = Play_ctr + 1
    WHERE Album_ID = v_albumID;
END;
```



	USER_ID	USERNAME	NAME	DOB	GENDER	PROFILE_PIC	ARTIST_ID	SUBSCRIPTION_START_DA
1	U005	indielover	Mia Thompson	21-03-95	F	https://some-image-link/mia_thompson.jpg	AR014	04-12-23
2	U007	retrovibes	Noah Brown	26-06-85	M	https://some-image-link/noah_brown.jpg	AR019	28-11-23
3	U013	jazzaddict	Emma Taylor	05-11-83	F	https://some-image-link/emma_taylor.jpg	AR012	05-10-23
4	U015	yogavibes	Chloe Nguyen	17-02-90	F	https://some-image-link/chloe_nguyen.jpg	AR002	04-08-23
5	U015	yogavibes	Chloe Nguyen	17-02-90	F	https://some-image-link/chloe_nguyen.jpg	AR002	15-10-23
6	U018	kids_bop	Daniel Anderson	20-05-02	M	https://some-image-link/daniel_anderson.jpg	AR003	26-06-23
7	U001	musiclover24	Sarah Johnson	12-05-96	F	https://some-image-link/sarah_johnson.jpg	AR001	15-02-24
8	U003	edm_enthusiast	Emily Davis	08-02-98	F	https://some-image-link/emily_davis.jpg	AR009	20-03-24
9	U013	jazzaddict	Emma Taylor	05-11-83	F	https://some-image-link/emma_taylor.jpg	AR012	06-11-23
10	U015	yogavibes	Chloe Nguyen	17-02-90	F	https://some-image-link/chloe_nguyen.jpg	AR002	16-11-23
11	U017	traveltunes	Ava Williams	30-01-96	F	https://some-image-link/ava_williams.jpg	AR007	01-12-23
12	U005	indielover	Mia Thompson	21-03-95	F	https://some-image-link/mia_thompson.jpg	AR014	05-01-24
13	U006	kpopexplorer	Sofia Lee	04-10-00	F	https://some-image-link/sofia_lee.jpg	AR006	29-02-24
14	U002	rockfanatic	Alex Miller	23-11-89	M	https://some-image-link/alex_miller.jpg	AR013	12-03-24
15	U009	classicalconnoisseur	Jacob Wilson	18-09-80	M	https://some-image-link/jacob_wilson.jpg	AR016	17-05-23
16	U011	tamil_musiclover	Priya Kumar	16-07-93	F	https://some-image-link/priya_kumar.jpg	AR015	22-01-24
17	U011	tamil_musiclover	Priya Kumar	16-07-93	F	https://some-image-link/priya_kumar.jpg	AR015	23-02-24
18	U020	punjabifan	Harpreet Singh	18-10-90	M	https://some-image-link/harpreet_singh.jpg	AR020	05-03-24
19	U007	retrovibes	Noah Brown	26-06-85	M	https://some-image-link/noah_brown.jpg	AR019	29-12-23
20	U004	hiphophead	David Kim	15-08-92	M	https://some-image-link/david_kim.jpg	AR005	11-09-23
21	U008	latinbeats	Isabella Rodriguez	12-01-99	F	https://some-image-link/isabella_rodriguez.jpg	AR018	(null)
22	U010	countryfan	Olivia Patel	29-04-91	F	https://some-image-link/olivia_patel.jpg	AR008	(null)
23	U012	bollywoodbuff	Rahul Sharma	01-12-97	M	https://some-image-link/rahul_sharma.jpg	AR017	(null)
24	U014	focus_beats	Lucas Chen	29-08-94	M	https://some-image-link/lucas_chen.jpg	AR004	(null)
25	U016	workoutwarrior	Ethan Tremblay	03-09-95	M	https://some-image-link/ethan_tremblay.jpg	AR010	(null)



NORMALISATION AND DEPENDENCIES

TABLE ARTIST:

```
CREATE TABLE Artist (
  Artist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
  Category VARCHAR(50),
  Likes INT NOT NULL,
  Play_ctr INT NOT NULL,
  Picture VARCHAR(150)
);
```

Functional Dependencies:

Artist_ID → Name ,Category , Likes ,Play_ctr ,Picture

Potential Pitfalls:

Update Anomalies:

- If an artist changes their Category, you would need to update this information in multiple rows, potentially leading to inconsistencies.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

We can introduce a new table `CATEGORY` which stores Category_names with reference to the Category_ID and modify the `ARTIST` table to store Category_ID and reference it as a foreign key to `CATEGORY` table.

Third Normal Form (3NF):

The Artist table likely doesn't need further normalization to reach 3NF. All non-key columns appear to be directly dependent on the primary key.

Decomposed Table (SQL*Plus):

```
CREATE TABLE Category (
    Category_ID INT PRIMARY KEY,
    Category_Name VARCHAR(50) NOT NULL
);

CREATE TABLE Artist (
    Artist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Category_ID INT NOT NULL,
    Likes INT NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    FOREIGN KEY (Category_ID) REFERENCES Category(Category_ID)
);
```

```
mysql> show tables;
+----------------+
| Tables_in_dbms |
+----------------+
| Artist         |
| Category       |
+----------------+
2 rows in set (0.00 sec)

mysql> desc Artist;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Artist_ID   | varchar(50)  | NO   | PRI | NULL    |       |
| Name        | varchar(50)  | NO   |     | NULL    |       |
| Category_ID | int          | NO   | MUL | NULL    |       |
| Likes       | int          | NO   |     | NULL    |       |
| Play_ctr    | int          | NO   |     | NULL    |       |
| Picture     | varchar(150) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> desc Category;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Category_ID   | int         | NO   | PRI | NULL    |       |
| Category_Name | varchar(50) | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

TABLE SONG:

CREATE TABLE Song (

Song_ID VARCHAR(50) NOT NULL PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Artist_ID VARCHAR(50),

Likes INT NOT NULL,

Play_ctr INT NOT NULL,

Picture VARCHAR(150),

Storage_Link VARCHAR(150) NOT NULL,

FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)

);

Functional Dependencies:

Song_ID → Name, Likes, Play_ctr, Picture, Storage_Link

Potential Pitfalls:

Update Anomalies:

- If an artist changes their Category, you would need to update this information in multiple rows, potentially leading to inconsistencies.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

The table is already in 2NF. There are no partial dependencies; all non-key columns depend on the entire primary key

Third Normal Form (3NF):

To reach 3NF and address the potential for redundancy with multiple artists, we have to create a many to many relation and modify the song table.

Decomposed Table (SQL*Plus):

```
CREATE TABLE Song_Artist (
   Song_ID VARCHAR(50) NOT NULL,
   Artist_ID VARCHAR(50) NOT NULL,
   PRIMARY KEY (Song_ID, Artist_ID),
   FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID),
```

```
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);


CREATE TABLE Song (
    Song_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Likes INT NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    Storage_Link VARCHAR(150) NOT NULL
);
```

```
mysql> show tables;
+----------------+
| Tables_in_dbms |
+----------------+
| Song           |
| Song_Artist    |
+----------------+
2 rows in set (0.00 sec)

mysql> desc Song;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| Song_ID      | varchar(50)  | NO   | PRI | NULL    |       |
| Name         | varchar(50)  | NO   |     | NULL    |       |
| Likes        | int          | NO   |     | NULL    |       |
| Play_ctr     | int          | NO   |     | NULL    |       |
| Picture      | varchar(150) | YES  |     | NULL    |       |
| Storage_Link | varchar(150) | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> desc Song_Artist;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Song_ID   | varchar(50) | NO   | PRI | NULL    |       |
| Artist_ID | varchar(50) | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

TABLE PLAYLIST:

```
CREATE TABLE Playlist (
```

```
    Playlist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Category VARCHAR(50),
    Created_At DATE NOT NULL,
    Picture VARCHAR(150)
);
```

Functional Dependencies:

Playlist_ID → Name, Category, Created_At, Picture

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key (Playlist_ID). There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Playlist;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Playlist_ID | varchar(50)  | NO   | PRI | NULL    |       |
| Name        | varchar(50)  | NO   |     | NULL    |       |
| Category    | varchar(50)  | YES  |     | NULL    |       |
| Created_At  | date         | NO   |     | NULL    |       |
| Picture     | varchar(150) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

TABLE PAYMENT_INFO:

CREATE TABLE Payment_Info (

Payment_ID VARCHAR(50) NOT NULL PRIMARY KEY,

User_ID VARCHAR(50) NOT NULL,

Payment_Method VARCHAR(50) NOT NULL,

Card_Number VARCHAR(20) NOT NULL,

Expiry_Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Customer(User_ID)

);

Functional Dependencies:

Payment_ID → User_ID, Payment_Method, Card_Number, Expiry_Date

Potential Pitfalls:

- **Data Consistency:** The foreign key relationship with the Customer table ensures referential integrity (a customer must exist for a payment to be associated with them). However, if a customer is deleted, their payment information will also need to be handled.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key (Payment_ID). There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Payment_Info;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| Payment_ID     | varchar(50) | NO   | PRI | NULL    |       |
| User_ID        | varchar(50) | NO   |     | NULL    |       |
| Payment_Method | varchar(50) | NO   |     | NULL    |       |
| Card_Number    | varchar(20) | NO   |     | NULL    |       |
| Expiry_Date    | date        | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

TABLE SUBSCRIPTION_HISTORY:

CREATE TABLE Subscription_History (

Subscription_ID VARCHAR(50) NOT NULL PRIMARY KEY,

User_ID VARCHAR(50) NOT NULL,

Plan_ID VARCHAR(50) NOT NULL,

Start_Date DATE NOT NULL,

```
  End_Date DATE,
  FOREIGN KEY (User_ID) REFERENCES Customer(User_ID),
  FOREIGN KEY (Plan_ID) REFERENCES Subscription_Plan(Plan_ID)
);
```

Functional Dependencies:

Subscription_ID → User_ID, Plan_ID, Start_Date, End_Date

Potential Pitfalls:

Data Consistency and Handling Changes:

- **Plan Changes:** What happens if a customer changes their Plan_ID mid-subscription? Would you create a new Subscription_History record, or update the existing one? Either choice presents challenges related to maintaining accurate subscription history.
- **Subscription Plan Deletion**: If a Subscription_Plan is removed from the system, the foreign key constraint in Subscription_History prevents you from simply deleting the plan.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Subscription_History;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Subscription_ID | varchar(50) | NO   | PRI | NULL    |       |
| User_ID         | varchar(50) | NO   |     | NULL    |       |
| Plan_ID         | varchar(50) | NO   |     | NULL    |       |
| Start_Date      | date        | NO   |     | NULL    |       |
| End_Date        | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

TABLE PLAY_HISTORY:

CREATE TABLE Play_History (

Play_ID VARCHAR(50) NOT NULL PRIMARY KEY,

User_ID VARCHAR(50) NOT NULL,

Song_ID VARCHAR(50) NOT NULL,

Play_Date TIMESTAMP NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Customer(User_ID),

FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)

);

<u>Functional Dependencies:</u>

Play_ID → User_ID, Song_ID, Play_Date

<u>Potential Pitfalls:</u>

Data Consistency

- **Orphaned Records:** If a user or a song is deleted from their respective tables (Customer or Song), the foreign key constraints in Play_History prevent the deletion and create orphaned records that lack referential integrity.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Play_History;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Play_ID   | varchar(50) | NO   | PRI | NULL    |       |
| User_ID   | varchar(50) | NO   |     | NULL    |       |
| Song_ID   | varchar(50) | NO   |     | NULL    |       |
| Play_Date | timestamp   | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

TABLE ALBUM

CREATE TABLE Album (

Album_ID VARCHAR(50) NOT NULL PRIMARY KEY,

```
    Name VARCHAR(50) NOT NULL,
    Artist_ID VARCHAR(50) NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);
```

TABLE ARTIST:

```
CREATE TABLE Artist (
   Artist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
   Name VARCHAR(50) NOT NULL,
   Category VARCHAR(50),
   Likes INT NOT NULL,
   Play_ctr INT NOT NULL,
   Picture VARCHAR(150)
);
```

Functional Dependencies:

Artist_ID → Name ,Category , Likes ,Play_ctr ,Picture

Potential Pitfalls:

Update Anomalies:

- If an artist changes their Category, you would need to update this information in multiple rows, potentially leading to inconsistencies.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

We can introduce a new table `CATEGORY` which stores Category_names with reference to the Category_ID and modify the `ARTIST` table to store Category_ID and reference it as a foreign key to `CATEGORY` table.

Third Normal Form (3NF):

The Artist table likely doesn't need further normalization to reach 3NF. All non-key columns appear to be directly dependent on the primary key.

<u>Decomposed Table (SQL*Plus):</u>

```
CREATE TABLE Category (
    Category_ID INT PRIMARY KEY,
    Category_Name VARCHAR(50) NOT NULL
);

CREATE TABLE Artist (
    Artist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Category_ID INT NOT NULL,
    Likes INT NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
```

FOREIGN KEY (Category_ID) REFERENCES Category(Category_ID)

);

```
mysql> show tables;
+----------------+
| Tables_in_dbms |
+----------------+
| Artist         |
| Category       |
+----------------+
2 rows in set (0.00 sec)

mysql> desc Artist;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Artist_ID   | varchar(50)  | NO   | PRI | NULL    |       |
| Name        | varchar(50)  | NO   |     | NULL    |       |
| Category_ID | int          | NO   | MUL | NULL    |       |
| Likes       | int          | NO   |     | NULL    |       |
| Play_ctr    | int          | NO   |     | NULL    |       |
| Picture     | varchar(150) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
6 rows in set (0.01 sec)

mysql> desc Category;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Category_ID   | int         | NO   | PRI | NULL    |       |
| Category_Name | varchar(50) | NO   |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

TABLE SONG:

CREATE TABLE Song (

Song_ID VARCHAR(50) NOT NULL PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Artist_ID VARCHAR(50),

Likes INT NOT NULL,

Play_ctr INT NOT NULL,

Picture VARCHAR(150),

```
  Storage_Link VARCHAR(150) NOT NULL,
  FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);
```

Functional Dependencies:

Song_ID → Name, Likes, Play_ctr, Picture, Storage_Link

Potential Pitfalls:

Update Anomalies:

- If an artist changes their Category, you would need to update this information in multiple rows, potentially leading to inconsistencies.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

The table is already in 2NF. There are no partial dependencies; all non-key columns depend on the entire primary key

Third Normal Form (3NF):

To reach 3NF and address the potential for redundancy with multiple artists, we have to create a many to many relation and modify the song table.

Decomposed Table (SQL*Plus):

```
CREATE TABLE Song_Artist (
  Song_ID VARCHAR(50) NOT NULL,
```

```
    Artist_ID VARCHAR(50) NOT NULL,
    PRIMARY KEY (Song_ID, Artist_ID),
    FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID),
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);


CREATE TABLE Song (
    Song_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Likes INT NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    Storage_Link VARCHAR(150) NOT NULL
);
```

```
mysql> show tables;
+----------------+
| Tables_in_dbms |
+----------------+
| Song           |
| Song_Artist    |
+----------------+
2 rows in set (0.00 sec)

mysql> desc Song;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| Song_ID      | varchar(50)  | NO   | PRI | NULL    |       |
| Name         | varchar(50)  | NO   |     | NULL    |       |
| Likes        | int          | NO   |     | NULL    |       |
| Play_ctr     | int          | NO   |     | NULL    |       |
| Picture      | varchar(150) | YES  |     | NULL    |       |
| Storage_Link | varchar(150) | NO   |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> desc Song_Artist;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Song_ID   | varchar(50) | NO   | PRI | NULL    |       |
| Artist_ID | varchar(50) | NO   | PRI | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

TABLE PLAYLIST:

```
CREATE TABLE Playlist (
    Playlist_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Category VARCHAR(50),
    Created_At DATE NOT NULL,
    Picture VARCHAR(150)
);
```

Functional Dependencies:

Playlist_ID → Name, Category, Created_At, Picture

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key (Playlist_ID). There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Playlist;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Playlist_ID | varchar(50)  | NO   | PRI | NULL    |       |
| Name        | varchar(50)  | NO   |     | NULL    |       |
| Category    | varchar(50)  | YES  |     | NULL    |       |
| Created_At  | date         | NO   |     | NULL    |       |
| Picture     | varchar(150) | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

TABLE PAYMENT_INFO:

CREATE TABLE Payment_Info (

Payment_ID VARCHAR(50) NOT NULL PRIMARY KEY,

User_ID VARCHAR(50) NOT NULL,

Payment_Method VARCHAR(50) NOT NULL,

Card_Number VARCHAR(20) NOT NULL,

Expiry_Date DATE NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Customer(User_ID)

);

Functional Dependencies:

Payment_ID → User_ID, Payment_Method, Card_Number, Expiry_Date

Potential Pitfalls:

- **Data Consistency:** The foreign key relationship with the Customer table ensures referential integrity (a customer must exist for a payment to be associated with them). However, if a customer is deleted, their payment information will also need to be handled.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key (Payment_ID). There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Payment_Info;
+----------------+-------------+------+-----+---------+-------+
| Field          | Type        | Null | Key | Default | Extra |
+----------------+-------------+------+-----+---------+-------+
| Payment_ID     | varchar(50) | NO   | PRI | NULL    |       |
| User_ID        | varchar(50) | NO   |     | NULL    |       |
| Payment_Method | varchar(50) | NO   |     | NULL    |       |
| Card_Number    | varchar(20) | NO   |     | NULL    |       |
| Expiry_Date    | date        | NO   |     | NULL    |       |
+----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

TABLE SUBSCRIPTION_HISTORY:

CREATE TABLE Subscription_History (

Subscription_ID VARCHAR(50) NOT NULL PRIMARY KEY,

User_ID VARCHAR(50) NOT NULL,

```
  Plan_ID VARCHAR(50) NOT NULL,
  Start_Date DATE NOT NULL,
  End_Date DATE,
  FOREIGN KEY (User_ID) REFERENCES Customer(User_ID),
  FOREIGN KEY (Plan_ID) REFERENCES Subscription_Plan(Plan_ID)
);
```

Functional Dependencies:

Subscription_ID → User_ID, Plan_ID, Start_Date, End_Date

Potential Pitfalls:

Data Consistency and Handling Changes:

- **Plan Changes:** What happens if a customer changes their Plan_ID mid-subscription? Would you create a new Subscription_History record, or update the existing one? Either choice presents challenges related to maintaining accurate subscription history.
- **Subscription Plan Deletion**: If a Subscription_Plan is removed from the system, the foreign key constraint in Subscription_History prevents you from simply deleting the plan.

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Subscription_History;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| Subscription_ID | varchar(50) | NO   | PRI | NULL    |       |
| User_ID         | varchar(50) | NO   |     | NULL    |       |
| Plan_ID         | varchar(50) | NO   |     | NULL    |       |
| Start_Date      | date        | NO   |     | NULL    |       |
| End_Date        | date        | YES  |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

TABLE PLAY_HISTORY:

CREATE TABLE Play_History (

Play_ID VARCHAR(50) NOT NULL PRIMARY KEY,

User_ID VARCHAR(50) NOT NULL,

Song_ID VARCHAR(50) NOT NULL,

Play_Date TIMESTAMP NOT NULL,

FOREIGN KEY (User_ID) REFERENCES Customer(User_ID),

FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)

);

<u>Functional Dependencies:</u>

Play_ID → User_ID, Song_ID, Play_Date

<u>Potential Pitfalls:</u>

Data Consistency

- **Orphaned Records:** If a user or a song is deleted from their respective tables (Customer or Song), the foreign key constraints in Play_History prevent the deletion and create orphaned records that lack referential integrity.

<u>Normalization Form:</u>

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

```
mysql> desc Play_History;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| Play_ID   | varchar(50) | NO   | PRI | NULL    |       |
| User_ID   | varchar(50) | NO   |     | NULL    |       |
| Song_ID   | varchar(50) | NO   |     | NULL    |       |
| Play_Date | timestamp   | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

TABLE Album:

CREATE TABLE Album (

```
    Album_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Artist_ID VARCHAR(50) NOT NULL,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);
```

Functional Dependencies:

- Album_ID -> Name, Play_ctr, Picture
- Artist_ID -> Name, Category, Likes

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

To achieve 2NF, we need to remove the partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

Decomposed Table (SQL*Plus):

```
CREATE TABLE Album_Details (
    Album_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Play_ctr INT NOT NULL,
    Picture VARCHAR(150),
```

```
    FOREIGN KEY (Album_ID) REFERENCES Album(Album_ID)
);

ALTER TABLE Album
    DROP COLUMN Play_ctr,
    DROP COLUMN Picture;
```

TABLE CUSTOMER:

```
CREATE TABLE Customer (
   User_ID VARCHAR(50) NOT NULL PRIMARY KEY,
   UserName VARCHAR(50) NOT NULL,
   Name VARCHAR(50) NOT NULL,
   DOB DATE NOT NULL,
   Gender VARCHAR(5) NOT NULL,
   Profile_Pic VARCHAR(150),
   Artist_ID VARCHAR(50) NOT NULL,
   FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID)
);
```

Functional Dependencies:

User_ID -> UserName, Name, DOB, Gender

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE LOGIN:

```
CREATE TABLE Login (
  UserName VARCHAR(50) NOT NULL PRIMARY KEY,
  Hashed_Password VARCHAR(256) NOT NULL,
  User_ID VARCHAR(50) NOT NULL,
  FOREIGN KEY (User_ID) REFERENCES Customer(User_ID)
);
```

Functional Dependencies:

- UserName -> Hashed_Password, User_ID

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE GENRE:

```
CREATE TABLE Genre (
    Genre_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL
);
```

Functional Dependencies:

Genre_ID -> Name

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE SUBSCRIPTION_PLAN:

```
CREATE TABLE Subscription_Plan (
```

```
    Plan_ID VARCHAR(50) NOT NULL PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Price DECIMAL(10,2) NOT NULL
);
```

Functional Dependencies:

`Plan_ID -> Name, Price`

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE PLAYLIST_SONG:

```
CREATE TABLE Playlist_Song (
   Playlist_ID VARCHAR(50) NOT NULL,
   Song_ID VARCHAR(50) NOT NULL,
   PRIMARY KEY (Playlist_ID, Song_ID),
   FOREIGN KEY (Playlist_ID) REFERENCES Playlist(Playlist_ID),
```

```
  FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)
);
```

Functional Dependencies:

- (Playlist_ID, Song_ID) -> (Since the primary key is composite, there's no additional data in this table)
- Playlist_ID -> Playlist Data (Data from the Playlist table)
- Song_ID -> Song Data (Data from the Song table)

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE ARTIST_GENRE:

```
CREATE TABLE Artist_Genre (
  Artist_ID VARCHAR(50) NOT NULL,
  Genre_ID VARCHAR(50) NOT NULL,
  PRIMARY KEY (Artist_ID, Genre_ID),
```

```
    FOREIGN KEY (Artist_ID) REFERENCES Artist(Artist_ID),
    FOREIGN KEY (Genre_ID) REFERENCES Genre(Genre_ID)
);
```

Functional Dependencies:

- (Artist_ID, Genre_ID) -> (Since the primary key is composite, there's no additional data in this table)
- Artist_ID -> Artist Data
- Genre_ID -> Genre Data

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE ALBUM_SONG:

```
CREATE TABLE Album_Song (
   Album_ID VARCHAR(50) NOT NULL,
   Song_ID VARCHAR(50) NOT NULL,
```

```
  PRIMARY KEY (Album_ID, Song_ID),
  FOREIGN KEY (Album_ID) REFERENCES Album(Album_ID),
  FOREIGN KEY (Song_ID) REFERENCES Song(Song_ID)
);
```

Functional Dependencies:

- (Album_ID, Song_ID) -> (Since the primary key is composite, there's no additional data in this table)
- Album_ID -> Album Data
- Song_ID -> Song Data

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

TABLE ADVERTISEMENT:

```
CREATE TABLE Advertisement (
  Ad_ID VARCHAR(50) NOT NULL PRIMARY KEY,
  Advertiser VARCHAR(50) NOT NULL,
  Duration INT NOT NULL,
  Ad_Link VARCHAR(150) NOT NULL,
  Start_Date DATE NOT NULL,
```

```
  End_Date DATE NOT NULL
);
```

Functional Dependencies:

Ad_ID -> Advertiser, Duration, Ad_Link, Start_Date, End_Date

Normalization Form:

First Normal Form (1NF):

The table is already in 1NF as there are no repeating groups of columns and each column has an atomic value.

Second Normal Form (2NF):

All non-key columns fully depend on the primary key. There are no partial dependencies.

Third Normal Form (3NF):

There are no transitive dependencies (non-key columns depending on other non-key columns) that would necessitate further normalization.

While in 3NF, you could consider separating the advertiser information into its own table:

Decomposed Table (SQL*Plus):

```
CREATE TABLE Advertisement (
  Ad_ID VARCHAR(50) NOT NULL PRIMARY KEY,
  Advertiser_ID VARCHAR(50) NOT NULL,
  Duration INT NOT NULL,
  Ad_Link VARCHAR(150) NOT NULL,
  Start_Date DATE NOT NULL,
  End_Date DATE NOT NULL,
  FOREIGN KEY (Advertiser_ID) REFERENCES Advertiser(Advertiser_ID)
);
```

```
CREATE TABLE Advertiser (
   Advertiser_ID VARCHAR(50) NOT NULL PRIMARY KEY,
   Advertiser VARCHAR(50) NOT NULL
);
```

CONCURRENT TRANSACTION

1. ADD SONG TO PLAYLIST CONCURRENT TRANSACTION

```
//User A's transaction
BEGIN TRANSACTION;
INSERT INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('My Playlist', 'Song123');
```

```
COMMIT;

// User B's transaction
BEGIN TRANSACTION;
INSERT INTO Playlist_Song (Playlist_ID, Song_ID) VALUES ('Workout Mix', 'Song456');
COMMIT;
```

2. UPDARE ARTIST LIKES CONCURRENT TRANSACTION

```
//User A's transaction
BEGIN TRANSACTION;
UPDATE Artist SET Likes = Likes + 1 WHERE Artist_ID = 'Artist123';
COMMIT;

// User B's transaction
BEGIN TRANSACTION;
UPDATE Artist SET Likes = Likes + 1 WHERE Artist_ID = 'Artist123';
COMMIT;
```

3. SUBSCRIBE TO PREMIUM PLAN CONCURRENT TRANSACTION

```
// User C's transaction
BEGIN TRANSACTION;
INSERT INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date)
VALUES ('SubsC', 'UserC', 'Premium', '2024-05-01');
COMMIT;
```

//User D's transaction

BEGIN TRANSACTION;

INSERT INTO Subscription_History (Subscription_ID, User_ID, Plan_ID, Start_Date) VALUES ('SubsD', 'UserD', 'Basic', '2024-05-01');

COMMIT;

4. DELETE SONG FROM PLAYLIST CONCURRENT TRANSACTION

//User E's transaction

BEGIN TRANSACTION;

DELETE FROM Playlist_Song WHERE Playlist_ID = 'My Playlist' AND Song_ID = 'SongXYZ';

COMMIT;

// User F's transaction

BEGIN TRANSACTION;

DELETE FROM Playlist_Song WHERE Playlist_ID = 'Favorites' AND Song_ID = 'Song789';

COMMIT;

5.ADD PLAY HISTORY CONCURRENT TRANSACTION

// User G's transaction

BEGIN TRANSACTION;

INSERT INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PlayG1', 'UserG', 'SongXYZ', CURRENT_TIMESTAMP);

COMMIT;

// User H's transaction

BEGIN TRANSACTION;

INSERT INTO Play_History (Play_ID, User_ID, Song_ID, Play_Date) VALUES ('PlayH1', 'UserH', 'Song456', CURRENT_TIMESTAMP);

COMMIT;

6. CREATE ADVERTISEMENT CONCURRENT TRANSACTION

//Admin A's transaction

BEGIN TRANSACTION;

INSERT INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('Ad123', 'ProductXYZ', 30, 'http://example.com/ad123', '2024-05-01', '2024-05-31');

COMMIT;

// Admin B's transaction

BEGIN TRANSACTION;

INSERT INTO Advertisement (Ad_ID, Advertiser, Duration, Ad_Link, Start_Date, End_Date) VALUES ('Ad456', 'ProductABC', 30, 'http://example.com/ad456', '2024-05-01', '2024-05-31');

COMMIT;

Chapter 4

SYSTEM REQUIREMENTS

1. **OPERATING SYSTEM**

   The SMS can be used on various operating systems, including Windows, macOS, and Linux. We can choose the one that we are most comfortable with. We recommend using Windows 10 or Windows 11.

2. **DEVELOPMENT ENVIRONMENT**

   We can use a variety of databases for creating a SMS, like MySQL, Oracle Database, etc. For our project we have chosen MySQL.

3. **HARDWARE**

   We would need a high-end computer for this service as it will have to handle audio encoding and processing at university scale. Possibly multiple PC connected together by a network

4. **NETWORK BANDWIDTH**

   We would require great bandwidth to process multiple request concurrently and eliminate the possibility of network lag and bad user experience

5. **STORAGE**

   We don't need much storage space for code. However, the assets (music and pictures) are going to be much greater in size and would require very much storage capacity with high availability

6. **LOGGING SERVER**

   To create reports and store data logs, we can use separate server with Microsoft Excel spreadsheets (.xlsx) and CSV files (Comma Separated Values, .csv) as file formats.

Chapter 5

USE OF DESIGN THINKING APPROACH

Design thinking is a human-centred approach to innovation that focuses on understanding the needs of users, generating creative solutions, and rapidly prototyping and testing ideas to solve complex problems. Here's how you could apply design thinking principles to the creation of an online music streaming platform:

1. Empathize: Understand User Needs

- Conduct user research to understand the preferences, behaviors, and pain points of music listeners.

- Use techniques such as interviews, surveys, and observation to gather insights into how people consume music, what features they value in a streaming platform, and what frustrations they encounter.

2. Define: Frame the Problem

- Synthesize the user research findings to identify key themes, challenges, and opportunities.

- Define the specific problems and needs that your music streaming platform aims to address, such as discovery, curation, personalization, social sharing, or accessibility.

3. Ideate: Generate Creative Solutions

- Brainstorm a wide range of ideas for features, functionalities, and experiences that could address the identified needs and problems.

- Encourage diverse perspectives and collaboration among team members to generate innovative solutions.

- Use techniques like brainstorming, mind mapping, and sketching to explore and visualize ideas.

4. Prototype: Build Quick, Low-Fidelity Solutions

- Create rapid prototypes or mock-ups of potential features and interfaces to bring your ideas to life.

- Use prototyping tools or even paper sketches to design and test concepts quickly and inexpensively.

- Iterate on your prototypes based on feedback from users and stakeholders, refining and improving them as you go.

5. Test: Gather Feedback and Iterate

- Conduct usability testing and gather feedback from real users to evaluate the effectiveness of your prototypes.

- Observe how users interact with your platform, identify pain points and areas for improvement, and gather insights into what works well and what doesn't.

- Use the feedback to refine your designs, make adjustments, and iterate on your prototypes, repeating the testing process as necessary.

6. Implement: Develop the Solution

- Once you've validated your design concepts through testing, begin developing the actual music streaming platform.

- Use agile development methodologies to incrementally build and release features, prioritizing those with the highest impact and value to users.

- Continuously gather feedback from users throughout the development process and incorporate it into your design decisions.

7. Launch: Release and Scale

- Launch the music streaming platform to the public, making sure to communicate its features, benefits, and value proposition effectively.

- Monitor user engagement, adoption, and feedback after the launch, and be prepared to make adjustments and updates as needed.

- Scale the platform to accommodate growing user demand, ensuring that it remains stable, performant, and user-friendly as it attracts more users.

Throughout the design thinking process, it's crucial to maintain a user-centric mindset, prioritizing the needs, preferences, and experiences of music listeners at every stage. By iterating on ideas, gathering feedback, and continuously refining your designs, you can create a compelling and successful online music streaming platform that delights users and meets their evolving needs.

Chapter 6

IMPLEMENTATION

Implementing an online streaming platform involves a series of steps to develop, deploy, and maintain the platform to ensure its functionality, scalability, and reliability. Here's an overview of the implementation process for an online streaming platform:

1. Requirement Analysis:

- Define the scope and objectives of the streaming platform.
- Gather requirements from stakeholders, including business goals, user needs, and technical specifications.
- Prioritize features and functionalities based on their importance and impact.

2. Technology Stack Selection:

- Choose the appropriate technology stack for the streaming platform, including backend, frontend, database, and hosting infrastructure.
- Consider factors such as scalability, performance, security, and development expertise when selecting technologies.

3. Architecture Design:

- Design the architecture of the streaming platform, including the overall system structure, data flow, and component interactions.
- Define the backend services, APIs, databases, and caching mechanisms required to support streaming, user authentication, content management, and other functionalities.
- Design the frontend components, including user interfaces, navigation flows, and media playback controls.

4. Development:

- Develop the backend components of the streaming platform, including user authentication, content management, recommendation algorithms, and payment processing.

- Implement the frontend interfaces and user experiences, ensuring responsiveness, accessibility, and cross-browser compatibility.

- Integrate third-party services and APIs for features such as content delivery, analytics, and social sharing.

- Implement security measures, including encryption, authentication, authorization, and data protection, to safeguard user data and platform integrity.

5. Content Management:

- Develop tools and workflows for content ingestion, encoding, storage, and management.

- Implement content delivery networks (CDNs) and caching mechanisms to optimize content delivery and reduce latency.

- Integrate digital rights management (DRM) solutions to protect copyrighted content and enforce licensing agreements.

6. Quality Assurance:

- Conduct rigorous testing of the streaming platform to identify and address bugs, errors, and performance issues.

- Perform functional testing, integration testing, regression testing, and performance testing to ensure the platform meets quality standards.

- Involve stakeholders, including users, in beta testing and usability testing to gather feedback and validate functionality.

7. Deployment:

- Prepare the streaming platform for deployment to production environments.

- Set up hosting infrastructure, including servers, databases, and networking configurations, to support the platform's scalability and reliability requirements.

- Deploy the platform to production environments using continuous integration and continuous deployment (CI/CD) pipelines to automate the deployment process and ensure consistency.

8. Monitoring and Optimization:

- Implement monitoring tools and systems to track the performance, availability, and usage of the streaming platform.

- Monitor key metrics such as server load, response times, error rates, and user engagement to identify areas for optimization and improvement.

- Continuously optimize the platform for scalability, performance, and cost-effectiveness by tuning configurations, optimizing code, and adopting best practices.

9. Maintenance and Support:

- Provide ongoing maintenance and support for the streaming platform, including bug fixes, security updates, and feature enhancements.

- Offer user support services, including helpdesk support, documentation, and tutorials, to assist users with platform usage and troubleshooting.

- Iterate on the platform based on user feedback, market trends, and technological advancements to ensure its continued relevance and competitiveness.

CODE

mainview.tsx

```
'use client';


import { Card, CardContent } from "@/components/ui/card";

import { Carousel, CarouselContent, CarouselItem, CarouselNext, CarouselPrevious } from "@/components/ui/carousel";

import { useEffect, useState } from "react";

import { fetchSongs } from '../utils/songsApi'; // Import from API file


export const MainView = () => {

  const [songs, setSongs] = useState([]);

  const [isLoading, setIsLoading] = useState(false);

  const [error, setError] = useState(null);


  useEffect(() => {

    const fetchData = async () => {

      setIsLoading(true);

      try {

        const data = await fetchSongs();

        setSongs(data);

      } catch (err) {

        setError(err);

      } finally {

        setIsLoading(false);

      }
```

```
    };

    fetchData();
  }, []);

  return (
    <>
      <p className="text-white text-2xl">Songs</p>
      {isLoading && <div>Loading songs...</div>}
      {error && <div>Error fetching songs: {error.message}</div>}
      {!isLoading && !error && (
        <div className=" w-4/5 h-40 m-10">
          <Carousel opts={{ align: "start" }} className="w-full max-w-md">
            <CarouselContent>
              {songs.map((song, index) => (
                <CarouselItem key={index} className="md:basis-1/2 lg:basis-1/3">
                  <div className="p-1">
                    <Card>
                    <CardContent className="flex aspect-square items-center justify-center p-6">
                        <span className="text-xl font-semibold">{song.Name}</span>
                      </CardContent>
                    </Card>
                  </div>
                </CarouselItem>
              ))}
            </CarouselContent>
```

```
                <CarouselPrevious />
                <CarouselNext />
            </Carousel>
        </div>
    )}
  </>
  );
};
```

sidebar.tsx

```
"use client";

import { Button } from "@/components/ui/button";
import { Avatar, AvatarFallback, AvatarImage } from "@/components/ui/avatar"


export const Sidebar = () => {

  return (
    <>
      <div className=" w-1/4 h-full flex flex-col items-center py-4 justify-around">
        <div className=" w-4/5 h-20 flex justify-center items-center">
            <Avatar>
                <AvatarImage src="https://github.com/shadcn.png" />
                <AvatarFallback>CN</AvatarFallback>
            </Avatar>
            <p className=" m-4 text-white">
```

```
                Shoaib Akhtar
            </p>
        </div>
        <div className=" flex-col w-4/5 justify-around">
            <Button variant="ghost" className="w-full text-white">Home</Button>
            <Button variant="ghost" className="w-full text-white">Playlist</Button>
            <Button variant="ghost" className="w-full text-white">Explore</Button>
        </div>
    </div>
  </>
 );
};
```

topbar.tsx

```
"use client";

import { Button } from "@/components/ui/button";
import { Input } from "@/components/ui/input"

export const Topbar = () => {

  return (
    <>
      <div className=" w-full h-16 flex justify-between items-center px-10 py-10">
        <Button variant="ghost" className=" text-white">Recommended</Button>
        <div className="flex items-center rounded-full bg-white dark:bg-gray-800 p-2 shadow-md
m-2">
```

```
                <Input
                    className="flex-1 rounded-full py-2 px-4 leading-none text-gray-800 dark:text-white
bg-transparent focus:outline-none"
                    placeholder="Search..."
                    type="text"
                />
                <SearchIcon className="w-5 h-5 text-gray-800 dark:text-white" />
            </div>
        </div>
    </>
  );
};

function SearchIcon(props) {
  return (
    <svg
        {...props}
        xmlns="http://www.w3.org/2000/svg"
        width="24"
        height="24"
        viewBox="0 0 24 24"
        fill="none"
        stroke="currentColor"
        strokeWidth="2"
        strokeLinecap="round"
        strokeLinejoin="round"
```

```
      >
        <circle cx="11" cy="11" r="8" />
        <path d="m21 21-4.3-4.3" />
      </svg>
    )
}
```

avatar.tsx

```
"use client"

import * as React from "react"
import * as AvatarPrimitive from "@radix-ui/react-avatar"

import { cn } from "@/lib/utils"

const Avatar = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Root>,
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Root>
>(({ className, ...props }, ref) => (
  <AvatarPrimitive.Root
    ref={ref}
    className={cn(
      "relative flex h-10 w-10 shrink-0 overflow-hidden rounded-full",
      className
    )}
    {...props}
  />
```

```
))
Avatar.displayName = AvatarPrimitive.Root.displayName


const AvatarImage = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Image>,
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Image>
>(({ className, ...props }, ref) => (
  <AvatarPrimitive.Image
    ref={ref}
    className={cn("aspect-square h-full w-full", className)}
    {...props}
  />
))
AvatarImage.displayName = AvatarPrimitive.Image.displayName


const AvatarFallback = React.forwardRef<
  React.ElementRef<typeof AvatarPrimitive.Fallback>,
  React.ComponentPropsWithoutRef<typeof AvatarPrimitive.Fallback>
>(({ className, ...props }, ref) => (
  <AvatarPrimitive.Fallback
    ref={ref}
    className={cn(
      "flex h-full w-full items-center justify-center rounded-full bg-muted",
      className
    )}
    {...props}
```

```
  />
))
AvatarFallback.displayName = AvatarPrimitive.Fallback.displayName


export { Avatar, AvatarImage, AvatarFallback }
```

button.tsx

```
import * as React from "react"
import { Slot } from "@radix-ui/react-slot"
import { cva, type VariantProps } from "class-variance-authority"


import { cn } from "@/lib/utils"


const buttonVariants = cva(
  "inline-flex items-center justify-center whitespace-nowrap rounded-md text-sm font-medium ring-offset-background transition-colors focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:pointer-events-none disabled:opacity-50",
  {
    variants: {
      variant: {
        default: "bg-primary text-primary-foreground hover:bg-primary/90",
        destructive:
          "bg-destructive text-destructive-foreground hover:bg-destructive/90",
        outline:
          "border border-input bg-background hover:bg-accent hover:text-accent-foreground",
        secondary:
          "bg-secondary text-secondary-foreground hover:bg-secondary/80",
```

```
        ghost: "hover:bg-accent hover:text-accent-foreground",
        link: "text-primary underline-offset-4 hover:underline",
      },
      size: {
        default: "h-10 px-4 py-2",
        sm: "h-9 rounded-md px-3",
        lg: "h-11 rounded-md px-8",
        icon: "h-10 w-10",
      },
    },
    defaultVariants: {
      variant: "default",
      size: "default",
    },
  }
)

export interface ButtonProps
  extends React.ButtonHTMLAttributes<HTMLButtonElement>,
    VariantProps<typeof buttonVariants> {
  asChild?: boolean
}

const Button = React.forwardRef<HTMLButtonElement, ButtonProps>(
  ({ className, variant, size, asChild = false, ...props }, ref) => {
    const Comp = asChild ? Slot : "button"
```

```
    return (
      <Comp
        className={cn(buttonVariants({ variant, size, className }))}
        ref={ref}
        {...props}
      />
    )
  }
)
Button.displayName = "Button"

export { Button, buttonVariants }
```

card.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

const Card = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn(
      "rounded-lg border bg-card text-card-foreground shadow-sm",
      className
```

```
    )}
    {...props}
  />
))
Card.displayName = "Card"


const CardHeader = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("flex flex-col space-y-1.5 p-6", className)}
    {...props}
  />
))
CardHeader.displayName = "CardHeader"


const CardTitle = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLHeadingElement>
>(({ className, ...props }, ref) => (
  <h3
    ref={ref}
    className={cn(
      "text-2xl font-semibold leading-none tracking-tight",
```

```
      className
    )}
    {...props}
  />
))
CardTitle.displayName = "CardTitle"


const CardDescription = React.forwardRef<
  HTMLParagraphElement,
  React.HTMLAttributes<HTMLParagraphElement>
>(({ className, ...props }, ref) => (
  <p
    ref={ref}
    className={cn("text-sm text-muted-foreground", className)}
    {...props}
  />
))
CardDescription.displayName = "CardDescription"


const CardContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div ref={ref} className={cn("p-6 pt-0", className)} {...props} />
))
CardContent.displayName = "CardContent"
```

```
const CardFooter = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => (
  <div
    ref={ref}
    className={cn("flex items-center p-6 pt-0", className)}
    {...props}
  />
))
CardFooter.displayName = "CardFooter"

export { Card, CardHeader, CardFooter, CardTitle, CardDescription, CardContent }
```

carousel.tsx

```
"use client"

import * as React from "react"
import useEmblaCarousel, {
  type UseEmblaCarouselType,
} from "embla-carousel-react"
import { ArrowLeft, ArrowRight } from "lucide-react"

import { cn } from "@/lib/utils"
import { Button } from "@/components/ui/button"
```

```
type CarouselApi = UseEmblaCarouselType[1]

type UseCarouselParameters = Parameters<typeof useEmblaCarousel>

type CarouselOptions = UseCarouselParameters[0]

type CarouselPlugin = UseCarouselParameters[1]


type CarouselProps = {

  opts?: CarouselOptions

  plugins?: CarouselPlugin

  orientation?: "horizontal" | "vertical"

  setApi?: (api: CarouselApi) => void

}


type CarouselContextProps = {

  carouselRef: ReturnType<typeof useEmblaCarousel>[0]

  api: ReturnType<typeof useEmblaCarousel>[1]

  scrollPrev: () => void

  scrollNext: () => void

  canScrollPrev: boolean

  canScrollNext: boolean

} & CarouselProps


const CarouselContext = React.createContext<CarouselContextProps | null>(null)


function useCarousel() {

  const context = React.useContext(CarouselContext)
```

```
  if (!context) {
    throw new Error("useCarousel must be used within a <Carousel />")
  }

  return context
}

const Carousel = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement> & CarouselProps
>(
  (
    {
      orientation = "horizontal",
      opts,
      setApi,
      plugins,
      className,
      children,
      ...props
    },
    ref
  ) => {
    const [carouselRef, api] = useEmblaCarousel(
      {
```

```
    ...opts,
    axis: orientation === "horizontal" ? "x" : "y",
  },
  plugins
)
const [canScrollPrev, setCanScrollPrev] = React.useState(false)
const [canScrollNext, setCanScrollNext] = React.useState(false)

const onSelect = React.useCallback((api: CarouselApi) => {
  if (!api) {
    return
  }

  setCanScrollPrev(api.canScrollPrev())
  setCanScrollNext(api.canScrollNext())
}, [])

const scrollPrev = React.useCallback(() => {
  api?.scrollPrev()
}, [api])

const scrollNext = React.useCallback(() => {
  api?.scrollNext()
}, [api])

const handleKeyDown = React.useCallback(
```

```
  (event: React.KeyboardEvent<HTMLDivElement>) => {
    if (event.key === "ArrowLeft") {
      event.preventDefault()
      scrollPrev()
    } else if (event.key === "ArrowRight") {
      event.preventDefault()
      scrollNext()
    }
  },
  [scrollPrev, scrollNext]
)

React.useEffect(() => {
  if (!api || !setApi) {
    return
  }

  setApi(api)
}, [api, setApi])

React.useEffect(() => {
  if (!api) {
    return
  }

  onSelect(api)
```

```
  api.on("reInit", onSelect)
  api.on("select", onSelect)

  return () => {
    api?.off("select", onSelect)
  }
}, [api, onSelect])

return (
  <CarouselContext.Provider
    value={{
      carouselRef,
      api: api,
      opts,
      orientation:
        orientation || (opts?.axis === "y" ? "vertical" : "horizontal"),
      scrollPrev,
      scrollNext,
      canScrollPrev,
      canScrollNext,
    }}
  >
    <div
      ref={ref}
      onKeyDownCapture={handleKeyDown}
      className={cn("relative", className)}
```

```
          role="region"
          aria-roledescription="carousel"
          {...props}
        >
          {children}
        </div>
      </CarouselContext.Provider>
    )
  }
)
Carousel.displayName = "Carousel"


const CarouselContent = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  const { carouselRef, orientation } = useCarousel()


  return (
    <div ref={carouselRef} className="overflow-hidden">
      <div
        ref={ref}
        className={cn(
          "flex",
          orientation === "horizontal" ? "-ml-4" : "-mt-4 flex-col",
          className
```

```
        )}
        {...props}
      />
    </div>
  )
})
CarouselContent.displayName = "CarouselContent"


const CarouselItem = React.forwardRef<
  HTMLDivElement,
  React.HTMLAttributes<HTMLDivElement>
>(({ className, ...props }, ref) => {
  const { orientation } = useCarousel()


  return (
    <div
      ref={ref}
      role="group"
      aria-roledescription="slide"
      className={cn(
        "min-w-0 shrink-0 grow-0 basis-full",
        orientation === "horizontal" ? "pl-4" : "pt-4",
        className
      )}
      {...props}
    />
```

```
  )
})
CarouselItem.displayName = "CarouselItem"


const CarouselPrevious = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<typeof Button>
>(({ className, variant = "outline", size = "icon", ...props }, ref) => {
  const { orientation, scrollPrev, canScrollPrev } = useCarousel()


  return (
    <Button
      ref={ref}
      variant={variant}
      size={size}
      className={cn(
        "absolute  h-8 w-8 rounded-full",
        orientation === "horizontal"
          ? "-left-12 top-1/2 -translate-y-1/2"
          : "-top-12 left-1/2 -translate-x-1/2 rotate-90",
        className
      )}
      disabled={!canScrollPrev}
      onClick={scrollPrev}
      {...props}
    >
```

```
      <ArrowLeft className="h-4 w-4" />
      <span className="sr-only">Previous slide</span>
    </Button>
  )
})
CarouselPrevious.displayName = "CarouselPrevious"


const CarouselNext = React.forwardRef<
  HTMLButtonElement,
  React.ComponentProps<typeof Button>
>(({ className, variant = "outline", size = "icon", ...props }, ref) => {
  const { orientation, scrollNext, canScrollNext } = useCarousel()


  return (
    <Button
      ref={ref}
      variant={variant}
      size={size}
      className={cn(
        "absolute h-8 w-8 rounded-full",
        orientation === "horizontal"
          ? "-right-12 top-1/2 -translate-y-1/2"
          : "-bottom-12 left-1/2 -translate-x-1/2 rotate-90",
        className
      )}
      disabled={!canScrollNext}
```

```
      onClick={scrollNext}
      {...props}
    >
      <ArrowRight className="h-4 w-4" />
      <span className="sr-only">Next slide</span>
    </Button>
  )
})
CarouselNext.displayName = "CarouselNext"

export {
  type CarouselApi,
  Carousel,
  CarouselContent,
  CarouselItem,
  CarouselPrevious,
  CarouselNext,
}
```

input.tsx

```
import * as React from "react"

import { cn } from "@/lib/utils"

export interface InputProps
  extends React.InputHTMLAttributes<HTMLInputElement> {}
```

```
const Input = React.forwardRef<HTMLInputElement, InputProps>(
  ({ className, type, ...props }, ref) => {
    return (
      <input
        type={type}
        className={cn(
          "flex h-10 w-full rounded-md border border-input bg-background px-3 py-2 text-sm ring-offset-background file:border-0 file:bg-transparent file:text-sm file:font-medium placeholder:text-muted-foreground focus-visible:outline-none focus-visible:ring-2 focus-visible:ring-ring focus-visible:ring-offset-2 disabled:cursor-not-allowed disabled:opacity-50",
          className
        )}
        ref={ref}
        {...props}
      />
    )
  }
)
Input.displayName = "Input"


export { Input }
```

cn.ts

```
import { ClassValue, clsx } from "clsx";
import { twMerge } from "tailwind-merge";
```

```
export function cn(...inputs: ClassValue[]) {
   return twMerge(clsx(inputs));
}
```

songsApi.js

```
// songsApi.js
export const fetchSongs = async () => {
   const response = await fetch('http://localhost:5000/songs'); // Adjust URL if needed
   if (!response.ok) {
      throw new Error('Failed to fetch songs');
   }
   return response.json();
};
```

page.tsx

```
import Image from "next/image";
import { BackgroundGradientAnimation } from "./components/ui/background-gradient-animation";
import { BackgroundGradientAnimationInternal } from "./components/ui/background-gradient-animation_internal";
import { Topbar } from "./components/topbar";
import { Sidebar } from "./components/sidebar";
import { MainView } from "./components/mainview";


export default function Home() {
  return (<>
   <BackgroundGradientAnimation className="flex justify-center items-center h-screen">
     <BackgroundGradientAnimationInternal className=" w-full h-full flex justify-normal -rotate-180">
```

```
        <Sidebar />
        <div className=" w-full h-full flex-col">
          <Topbar />
          <MainView />
        </div>
      </BackgroundGradientAnimationInternal>
    </BackgroundGradientAnimation>
  </>
  );
}
```

global.css

```
@tailwind base;
  @tailwind components;
  @tailwind utilities;

  @layer base {
    :root {
      --background: 0 0% 100%;
      --foreground: 222.2 84% 4.9%;

      --card: 0 0% 100%;
      --card-foreground: 222.2 84% 4.9%;

      --popover: 0 0% 100%;
      --popover-foreground: 222.2 84% 4.9%;
```

```
  --primary: 222.2 47.4% 11.2%;
  --primary-foreground: 210 40% 98%;

  --secondary: 210 40% 96.1%;
  --secondary-foreground: 222.2 47.4% 11.2%;

  --muted: 210 40% 96.1%;
  --muted-foreground: 215.4 16.3% 46.9%;

  --accent: 210 40% 96.1%;
  --accent-foreground: 222.2 47.4% 11.2%;

  --destructive: 0 84.2% 60.2%;
  --destructive-foreground: 210 40% 98%;

  --border: 214.3 31.8% 91.4%;
  --input: 214.3 31.8% 91.4%;
  --ring: 222.2 84% 4.9%;

  --radius: 0.5rem;
}

.dark {
  --background: 222.2 84% 4.9%;
  --foreground: 210 40% 98%;
```

```
    --card: 222.2 84% 4.9%;
    --card-foreground: 210 40% 98%;

    --popover: 222.2 84% 4.9%;
    --popover-foreground: 210 40% 98%;

    --primary: 210 40% 98%;
    --primary-foreground: 222.2 47.4% 11.2%;

    --secondary: 217.2 32.6% 17.5%;
    --secondary-foreground: 210 40% 98%;

    --muted: 217.2 32.6% 17.5%;
    --muted-foreground: 215 20.2% 65.1%;

    --accent: 217.2 32.6% 17.5%;
    --accent-foreground: 210 40% 98%;

    --destructive: 0 62.8% 30.6%;
    --destructive-foreground: 210 40% 98%;

    --border: 217.2 32.6% 17.5%;
    --input: 217.2 32.6% 17.5%;
    --ring: 212.7 26.8% 83.9%;
  }
}
```

```
  @layer base {
    * {
      @apply border-border;
    }
    body {
      @apply bg-background text-foreground;
    }
  }
```

BACKEND

connection.js

```
const mysql = require('mysql2/promise');


const pool = mysql.createPool({
    host: 'localhost',
    user: 'root',
    password: 'my-secret-pw',
    database: 'dbms'
});


module.exports = pool;
```

server.js

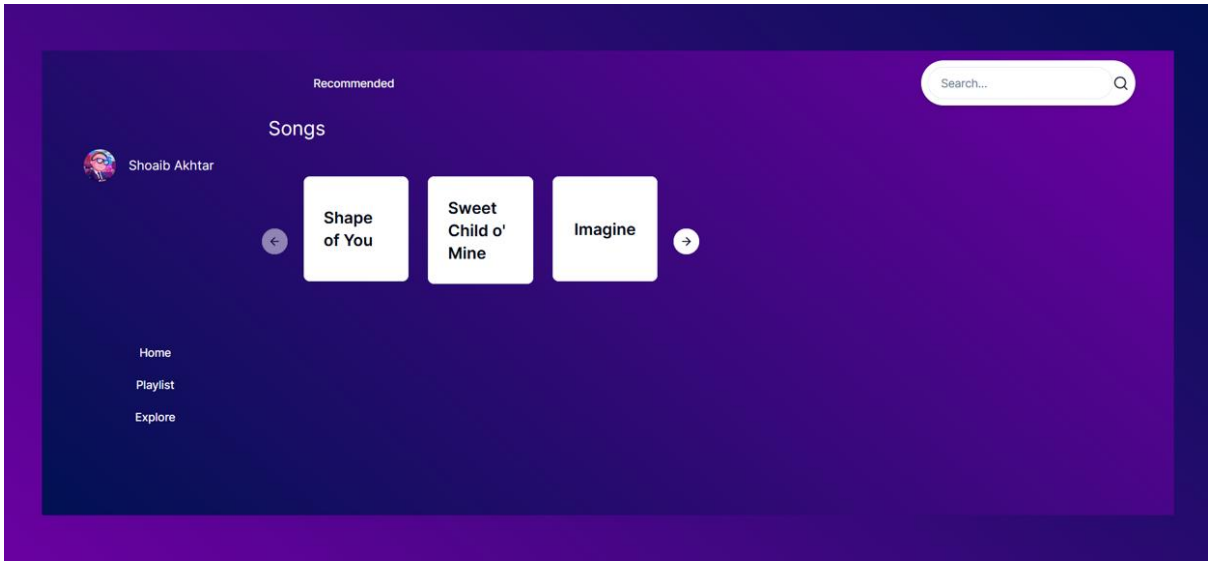
```
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const cors = require('cors');
```

```
const corsOptions = {
    origin: 'http://localhost:3000', // Allow requests from specific origin
    optionsSuccessStatus: 200 // Some legacy browsers choke on 204
};

app.use(cors(corsOptions));




// Import API routes
const artistRouter = require('./songs');
// ... other routers

// Middleware
app.use(bodyParser.json());

// API Route Mounting
app.use('/songs', artistRouter);
// ... other routes

// Start the server
const port = 5000;
app.listen(port, () => {
    console.log(Server listening on port ${port});
});
```

songs.js

```
const express = require('express');

const router = express.Router();

const pool = require('./connection');


// Get all artists

router.get('/', async (req, res) => {

    try {

        const [rows] = await pool.query('SELECT Name FROM Song');

        res.json(rows);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

});


module.exports = router;
```

RESULTS


Recommended
Search...
Songs
Shoaib Akhtar
Shape of You
Sweet Child o' Mine
Imagine
Home
Playlist
Explore

CONCLUSION

In conclusion, the development of our online music streaming platform represents a culmination of extensive research, user-centric design, iterative prototyping, and collaborative effort. Throughout the design thinking process, we prioritized understanding the diverse needs and preferences of music listeners, aiming to create an engaging and intuitive platform that enhances their music listening experience.

Through empathetic user research, we gained valuable insights into the behaviors, motivations, and pain points of our target audience. This informed our design decisions and guided the development of features and functionalities that address real user needs, from personalized recommendations and curated playlists to social sharing and seamless playback across devices.

Our iterative approach to design allowed us to rapidly prototype, test, and refine our ideas, ensuring that the final platform meets the highest standards of usability, accessibility, and enjoyment. By embracing feedback from users and stakeholders at every stage of the process, we iteratively improved the platform, incorporating valuable insights and addressing user concerns to create a truly user-centered experience.

The final presentation of our online streaming platform showcases not only the innovative features and elegant design but also the thoughtfulness and rigor that went into its creation. From the executive summary to the implementation plan, we have demonstrated the value proposition of the platform, its potential impact on users and the business, and the roadmap for future growth and innovation.

In summary, our online music streaming platform is not just a product but a solution that brings joy, inspiration, and connection to millions of music lovers around the world. It embodies the principles of design thinking, human-centered design, and continuous improvement, and we are excited to launch it into the world, knowing that it will enrich the lives of our users and contribute to the evolution of the music streaming industry.

REFERENCES

1. **GeeksForGeeks**
   https://www.geeksforgeeks.org/how-to-design-a-database-for-music-streaming-app/

2. **GITHUB-** GitHub - thealoneprogrammer/Musical-World: DBMS Mini Project that basically designed for online music player

DEVELOPMENT ENVIRONMENTS

- VirtualBox
- Ubuntu OS
- OnlineGDB Compiler for C
    - https://www.onlinegdb.com/online_c_compiler