WolfMedia

A Media Streaming Service

CSC 540 Database Management Systems

Project Report 2

Team W

Naveen Jayanna - njayann,
Samarth Purushothaman - spurush,
Srilekha Gudipati - sngudipa,
Sai Krishna Teja Varma Manthena - smanthe

March 12, 2023

Assumptions

a) Every Artist/ Podcast Host/ Record Label will have accounts.
b) AccountID is the username of the account, it must be unique.
c) Each artist is associated with exactly one record label.
d) Each song has a unique track ID within its album.
e) Each song is assigned to exactly one album (song cannot exist without an album)
f) Each song has a unique title within the album it resides.
g) Every Album belongs to a single main artist, other artists will be collaborators on the songs.
h) Every Podcast will have a Podcast Host
i) Podcast episode titles are unique within a podcast.
j) Each podcast episode is assigned to exactly one podcast (podcast episodes cannot exist without a podcast).
k) Subscribers pay a flat recurring fee every month (like Spotify) and can subscribe to any number of artists and/or podcasts (similar to YouTube).
l) Only those who have subscribed to the WolfMedia application can access the content within it.
m) Play count for each song and episode is incremented when a WolfMedia subscriber plays it.
n) Each podcast episode is paid out to the host(s) once by WolfMedia through a one-time payment.
o) The revenue generated by a podcast episode is evenly split amongst its hosts.
p) Multiple countries can speak one language and many languages can be spoken/listened to in one country.
q) Type in account table is an Enum of {‘admin’, ‘podcasthost’, ‘artist’, ‘user’, ‘recordlabel’
r) statusOfSubscription in User table is an Enum of {‘active’, ‘expired’}
s) The flat fee for a podcast episode is $250 and the bonus per advertisement is $100, the payment made to the podcast host per episode can be given by this formula: Flat Fee + Number of Advertisements * Bonus.
t) RoyaltyPaid attribute is not required as the payment for each month is tracked through the PaysArtist table.
u) All the users pay a base fee of $20 per month as a subscription fee for the WolfMedia.

I. Derive a global relational database schema.

Account (accountID, type, password, registrationDate)

accountID -> accountID, type, password, registrationDate.

The accountID is a unique identifier that determines all the attributes of this relation; hence, this FD holds. The LHS of this FD is a super key, hence, it is in BCNF, therefore it is in 3NF.

Genre (genreName)

genreName -> genreName

This relation has a single attribute representing various Genre Names for the songs and the podcasts; hence, it is in 3NF.

User (userAccountID, firstName, lastName, phone, email, statusOfSubscription)

userAccountID -> userAccountID, firstName, lastName, phone, email, statusOfSubscription

This functional dependency holds because each user will have a unique account ID with Wolfmedia. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

firstName, lastName -> userAccountID, firstName, lastName, phone, email, statusOfSubscription

It can be assumed that the combination of firstname and lastname as an identifier for all the attributes in this relation, however there is a possibility for two users to have the same first name and last name, hence, this FD does not hold.

Phone -> userAccountID, firstName, lastName, phone, email, statusOfSubscription

Email -> userAccountID, firstName, lastName, phone, email, statusOfSubscription

Since the phone/email can also be used as a unique identifier instead of accountID, these two FDs hold. As the left-hand side of both FDs are superkeys, this is in BCNF. Therefore, it is in 3NF.

RecordLabel (recordLabelAccountID, labelName)

recordLabelAccountID -> recordLabelAccountID, labelName

This functional dependency holds because each record label will have a unique account ID with Wolfmedia. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

labelName -> recordLabelAccountID, labelName

This functional dependency does not hold, although can be assumed, as two labels can have the same name.

Artist (<u>artistAccountID</u>, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabelAccountID)

artistAccountID -> artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabelAccountID

This functional dependency holds because each Artist has a unique ID to determine all the attributes of this relation. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

artistName -> artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabelAccountID

This functional dependency does not hold as two artists can have the same name.

artistName, recordLabelAccountID -> artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabelAccountID

This functional dependency does not hold as two artists can have the same name can have contract with the same record label.

Album (<u>albumID</u>, albumName, releaseYear, edition)

albumID -> albumID, albumName, releaseYear, edition

This functional dependency holds because each album has a unique ID to determine all the attributes of this relation. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

albumName, releaseYear, edition->albumID, albumName, releaseYear, edition

This functional dependency does not hold because there can be different albums with the same name, released year and edition.

Releases (<u>artistAccountID</u>, <u>albumID</u>)

artistAccountID, albumID -> artistAccountID, albumID

Since both the attributes are required to identify the album releases by the artists also, this is a two-attribute relation hence, this FD is in BCNF and hence in 3NF.

Song (<u>albumID</u>, <u>songTitle</u>, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber, mainArtist)

albumID, songTitle -> albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber, mainArtist

As a song is a weak entity and is uniquely identified within an album, both the albumID and songTitle are required to uniquely identify each song. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

songTitle, releaseDate -> albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber, mainArtist

This FD does not hold as two songs on the same date with the same name can be released as well.

mainArtist, songTitle -> albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber, mainArtist

This FD does not hold good as an artist can have the same song title across different albums.

SongGenre (genreName, songTitle, albumID)

genreName, songTitle, albumID -> genreName, songTitle, albumID

LHS is a superkey and no subset can identify the entries of the SongGenre relationship independently, hence the BCNF. Thereby, it is in 3NF and there are no other FDs on this schema.

Collaborators (artistAccountID, songTitle, albumID)

artistAccountID, songTitle, albumID -> artistAccountID, songTitle, albumID

LHS is a superkey and no subset can identify the entries of the Collaborators relationship independently, hence the BCNF. Thereby, it is in 3NF and there are no other FDs on this schema.

ArtistSubscribers (artistAccountID, userAccountID)

artistAccountID, userAccountID -> artistAccountID, userAccountID

LHS is a superkey and no subset can identify the entries of the ArtistSubscribers relationship independently, hence the BCNF. Thereby, it is in 3NF and there are no other FDs on this schema.

Podcast (podcastID, podcastName, language, country, episodeCount, rating, totalSubscribers)

podcastID -> podcastID, podcastName, language, country, episodeCount, rating, totalSubscribers

This functional dependency holds because each podcast will have a unique ID and all other podcast attributes can be determined from the podcastID. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF and there are no other FDs on this schema.

podcastName -> podcastID, podcastName, language, country, episodeCount, rating, totalSubscribers

Although this FD can be assumed to hold for the relation, there is a possibility for two podcasts to have same name, hence, this FD does not hold for the relation.

PodcastHost (<u>podcastHostAccountID</u>, firstName, lastName, phone, email, city)

podcastHostAccountID -> podcastHostAccountID, firstName, lastName, phone, email, city

This functional dependency holds because each podcast host will have a unique account ID with Wolfmedia. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

firstname, lastname -> podcastHostAccountID, firstName, lastName, phone, email, city

It can be assumed that the combination of firstname and lastname as an identifier for all the attributes in this relation, however there is a possibility for two podcast hosts to have the same first name and last name, hence, this FD does not hold.

phone -> podcastHostAccountID, firstName, lastName, phone, email, city

email -> podcastHostAccountID, firstName, lastName, phone, email, city

Since the phone/email can also be used as a unique identifier instead of accountID, these two FDs hold. As the left-hand side of the both FDs are superkeys, this is in BCNF. Therefore, it is in 3NF.

Hosts (<u>podcastHostAccountID</u>, <u>podcastID</u>)

podcastHostAccountID, podcastID -> podcastHostAccountID, podcastID

LHS is a superkey and no subset can identify the entries of the Hosts relationship independently, hence the BCNF. Thereby, it is in 3NF and there are no other FDs on this schema.

PodcastEpisode (<u>podcastID</u>, <u>episodeTitle</u>, duration, releaseDate, listeningCount, advertisementCount)

podcastID, episodeTitle -> podcastID, episodeTitle, duration, releaseDate, listeningCount, advertisementCount

As podcast episode is a weak entity and is uniquely identified within a podcast, both the podcastID and episodeTitle are required to uniquely identify each episode. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

releaseDate, episodeTitle -> podcastID, episodeTitle, duration, releaseDate, listeningCount, advertisementCount

This FD does not hold as two episodes on the same date with the same name can be released as well.

PodcastGenre (<u>podcastID</u>, <u>genreName</u>)

podcastID, genreName -> podcastID, genreName

Since both the attributes are required to tag all relevant genres to every podcast, this FD is in BCNF and hence in 3NF. No other FDs hold on this table.

PodcastSubscribers (<u>podcastID</u>, <u>userAccountID</u>, rating)

podcastID, userAccountID -> podcastID, userAccountID, rating

This FD holds because we need the podcastID and the subscriber ID (which is the user account ID) in the subscribers table. Since the left-hand side of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

podcastID -> podcastID, userAccountID, rating

userAccountID -> podcastID, userAccountID, rating

These will not hold as multiple users can rate a podcast and one user be subscribed to or rate multiple podcasts.

Sponsor (<u>sponsorID</u>, sponsorName)

sponsorID -> sponsorID, sponsorName

Per definition of SponsorID that we have created, it is unique and is enough to identify all the other attributes of the Sponsor table. Since the LHS of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

sponsorName -> sponsorID, sponsorName

This will not hold as there could be two sponsors with the same name.

AffiliatedWith (<u>sponsorID</u>, <u>podcastID</u>, amount)

sponsorID, podcastID-> sponsorID, podcastID, amount

Since, sponsorID and podcastID is the unique identifier for the relation, and LHS of this FD is a superkey, this is in BCNF, hence, it is in 3NF.

UserPayment (<u>paymentID</u>, userAccountID, paymentDate)

paymentID -> paymentID, userAccountID, paymentDate

userAccountID, paymentDate -> paymentID, userAccountID, paymentDate

Both the FDs can identify a record in this relation to record a payment by a user. The LHS of this FD is a super key, hence, it is in BCNF, therefore, it is in 3NF.

PaysLabel (<u>paymentID</u>, songTitle, albumID, recordLabelAccountID, paymentDate, playCount, amount)

paymentID -> paymentID, songTitle, albumID, recordLabelAccountID, paymentDate, playCount, amount

songTitle, albumID, recordLabelAccountID, paymentDate -> paymentID, songTitle, albumID, recordLabelAccountID, paymentDate, playCount, amount

Both the FDs can identify a record in this relation to record a payment to the record label. The LHS of this FD is a super key, hence, it is in BCNF, therefore, it is in 3NF.

PaysArtist (<u>paymentID</u>, artistAccountID, recordLabelAccountID, songTitle, albumID, paymentDate, amount)

paymentID -> paymentID, artistAccountID, recordLabelAccountID, songTitle, albumID, paymentDate, amount

artistAccountID, recordLabelAccountID, songTitle, albumID, paymentDate -> paymentID, artistAccountID, recordLabelAccountID, songTitle, albumID, paymentDate, amount

Both the FDs can identify a record in this relation to record a payment to the artist from the record label. The LHS of this FD is a super key, hence, it is in BCNF, therefore, it is in 3NF.

PaysHost (<u>paymentID</u>, podcastHostAccountID, podcastID, episodeTitle, paymentDate, amount)

paymentID -> paymentID, podcastHostAccountID, podcastID, episodeTitle, paymentDate, amount

podcastHostAccountID, podcastID, episodeTitle, paymentDate -> paymentID, podcastHostAccountID, podcastID, episodeTitle, paymentDate, amount

Both the FDs can identify a record in this relation to record a payment to the host of the podcast. The LHS of this FD is a super key, hence, it is in BCNF, therefore, it is in 3NF.

SpecialGuest (<u>guestID</u>, guestName)

guestID -> guestID, guestName

Per definition of guestID that we have created, it is unique and is enough to identify all the other attributes of the Sponsor table. Since the LHS of the FD is a superkey, this is in BCNF. Therefore, it is in 3NF.

guestName -> guestID, guestName

This will not hold as there could be two guests with the same name.

Invites (<u>guestID</u>, <u>podcastID</u>, <u>episodeTitle</u>)

guestID, podcastID, episodeTitle -> guestID, podcastID, episodeTitle

The LHS of this FD is a super key identifying every record of the relation to describe the guests invited in an episode of a podcast, and it is the super key, hence it is in BCNF, therefore, it is in 3NF.

II. Design for global schema

The design for the relational schema is converting the global E/R diagrams to the schema using the E/R technique to convert the hierarchy of account to admin, podcast host, artist and user. This method reduces the NULL values and redundancies in the tables.

By incorporating the key of the one as an attribute of the many, we were able to merge the many-one relationships into attributes. As a result, there is less duplication and overhead from having several tables. Moreover, it expedites queries. For example, the 'contractedTo' relation is a many-to-one relation. So, we have removed it by adding the recordLabel attribute to the Artist table.

In our schema, every other relationship has been transformed into a relation. The keys to the entities they represent are contained in their attributes in the schema.

Account (accountID, type, password, registrationDate)

- accountID is the primary key and it is enforced to be NOT NULL.
- type, password, and registrationDate are required attributes for the Account table and should be enforced to be NOT NULL.

Genre (genreName)

- genreName is the primary key and it is enforced to be NOT NULL.

User (userAccountID, firstName, lastName, phone, email, statusOfSubscription)

- userAccountID is the primary key and is enforced to be NOT NULL.
- userAccountID is also the foreign key that references accountID from the Account table and follows cascading referential integrity on both delete and update operations.
- firstName, lastName, phone, email, city is required attributes for the User table and should be enforced to be NOT NULL.
- Phone and email attributes should be UNIQUE.

RecordLabel (recordLabelAccountID, labelName)

- recordLabelAccountID is the primary key and is enforced to be NOT NULL.
- recordLabelAccountID is also the foreign key that references accountID from Account table and follows cascading referential integrity on both delete and update operations.
- labelName is a required attribute for the recordLabel and should be enforced to be NOT NULL.

Artist (artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabelAccountID)

- artistAccountID is the primary key and is enforced to be NOT NULL.
- artistAccountID is a foreign key that references accountID from Account table and follows cascading referential integrity on both delete and update operations.
- firstName, lastName, phone, email, city, monthlyListeners are required attributes for the Artist table and should be enforced to be NOT NULL. monthlyListeners has a default value of 0.

- recordLabelAccountID is a foreign key that references recordLabelAccountID from RecordLabel table and follows cascading referential integrity for update operation and SET NULL for delete operation.
- primaryGenre is a foreign key that references genreName from Genre table and follows cascading referential integrity for update operation and SET NULL for delete operation.
- recordLabelAccountID can be NULL which means that Artist is currently not contracted to any Record Label and primaryGenre can be NULL which means that the Artist currently has no primaryGenre.

Album (albumID, albumName, releaseYear, edition)

- albumID is the primary key and it is enforced to be NOT NULL.
- albumtName, releaseYear, edition is required attributes for the Album table and should be enforced to be NOT NULL.

Releases (artistAccountID, albumID)

- artistAccountID, albumID is the primary key and is enforced to be NOT NULL.
- artistAccountID is also the foreign key that references artistAccountID from the Artist table.
- albumID is also the foreign key that references albumID from the Album table.
- The foreign keys artistAccountID and albumID follow cascading referential integrity on both delete and update operations.

Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber, mainArtist)

- albumID, songTitle is the primary key and is enforced to be NOT NULL.
- albumID is a foreign key that references albumID from Album table and follows cascading referential integrity on both delete and update operations.
- mainArtist is a foreign key that references accountID from the Account table and follows cascading referential integrity for update operation and SET NULL for delete operation.
- mainArtist can be NULL which means that the song currently does not have a main artist and a main artist needs to be assigned immediately.
- duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber are required attributes for the Song table and should be enforced to be NOT NULL. playCount has a default value of 0.

SongGenre (genreName, songTitle, albumID)

- genreName, songTitle, albumID is the primary key and is enforced to be NOT NULL.
- genreName is also the foreign key that references genreName from the Genre table.
- songTitle, albumID is also the foreign key that references songTitle, albumID from the Song table.
- The foreign keys genreName and songTitle, albumID follow cascading referential integrity on both delete and update operations.

Collaborators (artistAccountID, songTitle, albumID)

- artistAccountID, songTitle, albumID is the primary key and is enforced to be NOT NULL.
- artistAccountID is also the foreign key that references artistAccountID from the Artist table.
- songTitle, albumID is also the foreign key that references songTitle, albumID from the Song table.
- The foreign keys artistAccountID, songTitle, and albumID follow cascading referential integrity on both delete and update operations.

ArtistSubscribers (artistAccountID, userAccountID)

- artistAccountID, userAccountID is the primary key and is enforced to be NOT NULL.
- artistAccountID is a foreign key that references artistAccountID from the Artist table.
- userAccountID is also a foreign key that references userAccountID from the User table.
- The foreign keys artistAccountID and userAccountID follow cascading referential integrity on both delete and update operations.

Podcast (podcastID, podcastName, language, country, episodeCount, rating, totalSubscribers)

- podcastID is the primary key and it is enforced to be NOT NULL.
- podcastName, language, country, episodeCount are required attributes for the Podcast table and should be enforced to be NOT NULL.
- Rating and total subscribers can be null, and they will be calculated based on the entries made in the PodcastSubscribers table.

PodcastHost (podcastHostAccountID, firstName, lastName, phone, email, city)

- podcastHostAccountID is the primary key and is enforced to be NOT NULL.
- podcastHostAccountID is also the foreign key that references accountID from the Account table and follows cascading referential integrity on both delete and update operations.
- firstName, lastName, phone, email, city is required attributes for the PodcastHost table and should be enforced to be NOT NULL.
- Phone and email are UNIQUE attributes.

Hosts (podcastHostAccountID, podcastID)

- podcastHostAccountID, podcastID is the primary key and is enforced to be NOT NULL.
- podcastHostAccountID is also the foreign key that references accountID from the Account table.
- podcastID is also the foreign key that references podcastID from the Podcast table.
- The foreign keys podcastHostAccountID and podcastID follow cascading referential integrity on both delete and update operations.

PodcastEpisode (podcastID, episodeTitle, duration, releaseDate, listeningCount, advertisementCount)

- podcastID, episodeTitle is the primary key and is enforced to be NOT NULL.
- podcastID is also the foreign key that references podcastID from Podcast table and follows cascading referential integrity on both delete and update operations.

- duration, releaseDate, listeningCount, advertisementCount are required attributes for the PodcastEpisode table and should be enforced to be NOT NULL.

PodcastGenre (podcastID, genreName)

- podcastID, genreName is the primary key and is enforced to be NOT NULL.
- podcastID is also the foreign key that references podcastID from Podcast table and follows cascading referential integrity on both delete and update operations.

PodcastSubscribers (podcastID, userAccountID, rating)

- podcastID, userAccountID is the primary key and both values are enforced to be NOT NULL.
- userAccountID is also the foreign key that references accountID from the Account table.
- The foreign key userAccountID follows cascading referential integrity on both delete and update operations.
- Rating can be null initially which means a given user hasn`t rated a given podcast. It can be updated when the user wants to give a rating.

Sponsor (sponsorID, sponsorName)

- sponsorID is the primary key and is enforced to be NOT NULL.
- sponsorName is a required attribute for the Sponsor table and should be enforced to be NOT NULL.

AffiliatedWith (sponsorID, podcastID, amount)

- sponsorID, podcastID is the primary key and both attributes are enforced to be NOT NULL.
- podcastID is also the foreign key that references podcastID from Podcast table and follows cascading referential integrity on both delete and update operations.
- sponsorID is also the foreign key that references sponsorID from the Sponsor table and follows cascading referential integrity on both delete and update operations.
- amount is a required attribute for the AffiliatedWith table and should be enforced to be NOT NULL.

UserPayment (paymentID, userAccountID, paymentDate)

- paymentID is the primary key and all the attributes are enforced to be NOT NULL.
- userAccountID is also the foreign key that references accountID from the Account table and follows set null on delete and cascade on update.
- userAccountID can be NULL in order to record all the payments from Users that have been deleted.
- paymentDate is a required attribute for the UserPayment table and should be enforced to be NOT NULL.

PaysLabel (paymentID, songTitle, albumID, recordLabelAccountID, paymentDate, playCount, amount)

- paymentID is the primary key and all the attributes are enforced to be NOT NULL.
- songTitle, albumID are foreign key references to songTitle and albumID from `Song` table and follows set null on delete and cascade on update.

- songTitle, albumID can be NULL in order to record all the payments to the record label for songs that have been deleted.
- recordLabelAccountID is a foreign key that references recordLabelAccountID from RecordLabel table and follows cascading referential integrity for both delete and update operations.
- recordLabelAccountID, playCount, amount is a required attribute for the PaysLabel table and should be enforced to be NOT NULL.

PaysArtist (paymentID, artistAccountID, recordLabelAccountID, songTitle, albumID, paymentDate, amount)

- paymentID is the primary key and all the attributes are enforced to be NOT NULL.
- songTitle, albumID are foreign key references to songTitle and albumID from `Song` table and follows set null on delete and cascade on update.
- recordLabelAccountID is a foreign key that references recordLabelAccountID from RecordLabel table and follows set null on delete and cascade on update.
- songTitle, albumID, recordLabelAccountID can be NULL in order to record all the payments to the artists for songs/record labels that have been deleted.
- artistAccountID is a foreign key that references artistAccountID from Artist table and follows cascading referential integrity for both delete and update operations.
- artistAccountID, paymentDate, amount is required attributes for the PaysArtist table and should be enforced to be NOT NULL.

PaysHost (paymentID, podcastHostAccountID, podcastID, episodeTitle, paymentDate, amount)

- paymentID is the primary key and all the attributes are enforced to be NOT NULL.
- podcastID, episodeTitle are foreign key references to podcastID, episodeTitle from `PodcastEpisode` table and follows set null on delete and cascade on update.
- podcastID, episodeTitle can be NULL in order to record all the payments to the Podcast Host for episodes that have been deleted.
- podcastHostAccountID is a foreign key that references podcastHostAccountID from PodcastHost table and follows cascading referential integrity for both delete and update operations.
- podcastHostAccountID, paymentDate, amount is required attributes for the PaysHost table and should be enforced to be NOT NULL.

SpecialGuest (guestID, guestName)

- guestID is the primary key and is enforced to be NOT NULL.
- guestName is a required attribute for the SpecialGuest and should be enforced to be NOT NULL.

Invites (guestID, podcastID, episodeTitle)

- guestID, podcastID, episodeTitle is the primary key and all the attributes are enforced to be NOT NULL.
- podcastID, episodeTitle is also the foreign key that references podcastID, episodeTitle from podcastEpisode, and follows cascading referential integrity on both delete and update operations.

III. Base relations

Account (accountID, type, password, registrationDate)

```
CREATE TABLE IF NOT EXISTS `Account` (
`accountID` INT NOT NULL AUTO_INCREMENT,
`type` VARCHAR(50) NOT NULL,
`password` VARCHAR(100) NOT NULL,
`registrationDate` DATE NOT NULL,
PRIMARY KEY(`accountID`)
);
```

Genre (genreName)

```
CREATE TABLE IF NOT EXISTS `Genre` (
`genreName` VARCHAR(50),
PRIMARY KEY(`genreName`)
);
```

User (userAccountID, firstName, lastName, phone, email, statusOfSubscription)

```
CREATE TABLE IF NOT EXISTS `User` (
`userAccountID` INT,
`firstName` VARCHAR(50) NOT NULL,
`lastName` VARCHAR(50) NOT NULL,
`phone` VARCHAR(30) NOT NULL,
`email` VARCHAR(50) NOT NULL,
`statusOfSubscription` VARCHAR(30) NOT NULL,
PRIMARY KEY(`userAccountID`),
FOREIGN KEY(`userAccountID`) REFERENCES
Account(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

RecordLabel (recordLabelAccountID, labelName)

```
CREATE TABLE IF NOT EXISTS `RecordLabel` (
`recordLabelAccountID` INT NOT NULL,
`labelName` VARCHAR(30) NOT NULL,
PRIMARY KEY (`recordLabelAccountID`),
FOREIGN KEY (`recordLabelAccountID`) REFERENCES
Account(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE);
```

Artist (artistAccountID, artistName, status, type, country, primaryGenre, monthlyListeners, recordLabelAccountID)

```
CREATE TABLE `Artist` ( `artistAccountID` INT NOT NULL,
`artistName` VARCHAR(50) NOT NULL,
`status` VARCHAR(50),
`type` VARCHAR(50),
```

```
`country` VARCHAR(50),
`primaryGenre` VARCHAR(50),
`monthlyListeners` INT DEFAULT 0,
`recordLabelAccountID` INT,
PRIMARY KEY (`artistAccountID`),
FOREIGN KEY (`artistAccountID`) REFERENCES
`Account`(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (`recordLabelAccountID`) REFERENCES
`RecordLabel`(`recordLabelAccountID`) ON DELETE SET NULL ON
UPDATE CASCADE)
FOREIGN KEY (`primaryGenre`) REFERENCES
`Genre`(`genreName`) ON DELETE SET NULL ON UPDATE CASCADE;
```

Album (albumID, albumName, releaseYear, edition)

```
CREATE TABLE `Album` (
`albumID` INT NOT NULL AUTO_INCREMENT,
`albumName` VARCHAR(255) NOT NULL,
`releaseYear` INT NOT NULL,
`edition` VARCHAR(255) NOT NULL,
PRIMARY KEY (`albumID`) );
```

Releases (artistAccountID, albumID)

```
CREATE TABLE `Releases` (
`artistAccountID` INT NOT NULL,
`albumID` INT NOT NULL,
PRIMARY KEY (`artistAccountID`, `albumID`),
FOREIGN KEY (`artistAccountID`) REFERENCES `Artist`
(`artistAccountID`) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (`albumID`) REFERENCES `Album` (`albumID`) ON
UPDATE CASCADE ON DELETE CASCADE );
```

Song (albumID, songTitle, duration, playCount, releaseDate, releaseCountry, language, royaltyRate, trackNumber, mainArtist)

```
CREATE TABLE `Song` (
`albumID` INT NOT NULL,
`songTitle` VARCHAR(50) NOT NULL,
`duration` INT NOT NULL,
`playCount` INT NOT NULL,
`releaseDate` DATE NOT NULL,
`releaseCountry` VARCHAR(50) NOT NULL,
`language` VARCHAR(50) NOT NULL,
`royaltyRate` DECIMAL(5,2) NOT NULL,
`royaltyPaid` DECIMAL(10,2) NOT NULL,
`trackNumber` INT NOT NULL,
`mainArtist` INT,
PRIMARY KEY (`albumID`, `songTitle`),
```

```
FOREIGN KEY (`albumID`) REFERENCES `Album`(`albumID`) ON
DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (`mainArtist`) REFERENCES
`Account`(`accountID`) ON DELETE SET NULL ON UPDATE CASCADE
);
```

SongGenre (genreName, songTitle, albumID)

```
CREATE TABLE IF NOT EXISTS `SongGenre` (
`genreName` VARCHAR(50) NOT NULL,
`albumID` INT NOT NULL,
`songTitle` VARCHAR(50) NOT NULL,
PRIMARY KEY (`genreName`, `albumID`, `songTitle`),
FOREIGN KEY (`genreName`) REFERENCES `Genre`(`genreName`)
ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (`albumID`, `songTitle`) REFERENCES
`Song`(`albumID`, `songTitle`)
);
```

Collaborators (artistAccountID, songTitle, albumID)

```
CREATE TABLE IF NOT EXISTS `Collaborators` (
`artistAccountID` INT NOT NULL,
`albumID` INT NOT NULL,
`songTitle` VARCHAR(50) NOT NULL,
PRIMARY KEY (`artistAccountID`, `albumID`, `songTitle`),
FOREIGN KEY (`artistAccountID`) REFERENCES
Artist`(`artistAccountID`) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY (`albumID`, `songTitle`) REFERENCES
`Song`(`albumID`, `songTitle`)
);
```

ArtistSubscribers (artistAccountID, userAccountID)

```
CREATE TABLE IF NOT EXISTS `ArtistSubscribers` (
`artistAccountID` INT NOT NULL,
`userAccountID` INT NOT NULL,
PRIMARY KEY (`artistAccountID`, `userAccountID`),
FOREIGN KEY (`artistAccountID`) REFERENCES
`Artist`(`artistAccountID`) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY (`userAccountID`) REFERENCES
`User`(`userAccountID`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Podcast (podcastID, podcastName, language, country, episodeCount, rating, totalSubscribers)

```
CREATE TABLE IF NOT EXISTS `Podcast` (
`podcastID` INT,
`podcastName` VARCHAR(50) NOT NULL,
```

```
`language` VARCHAR(10) NOT NULL,
`country` VARCHAR(20) NOT NULL,
`episodeCount` INT NOT NULL,
`rating` DECIMAL(10,2),
`totalSubscribers` INT,
PRIMARY KEY(`podcastID`)
);
```

PodcastHost (podcastHostAccountID, firstName, lastName, phone, email, city)

```
CREATE TABLE IF NOT EXISTS `PodcastHost` (
`podcastHostAccountID` INT,
`firstName` VARCHAR(50) NOT NULL,
`lastName` VARCHAR(50) NOT NULL,
`phone` VARCHAR(30) NOT NULL,
`email` VARCHAR(50) NOT NULL,
`city` VARCHAR(50) NOT NULL,
PRIMARY KEY(`podcastHostAccountID`),
FOREIGN KEY(`podcastHostAccountID`) REFERENCES
Account(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Hosts (podcastHostAccountID, podcastID)

```
CREATE TABLE IF NOT EXISTS `Hosts` (
`podcastHostAccountID` INT,
`podcastID` INT,
PRIMARY KEY(`podcastHostAccountID`, `podcastID`),
FOREIGN KEY(`podcastHostAccountID`) REFERENCES
Account(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(`podcastID`) REFERENCES Podcast(`podcastID`) ON
DELETE CASCADE ON UPDATE CASCADE
);
```

PodcastEpisode (podcastID, episodeTitle, duration, releaseDate, listeningCount, advertisementCount)

```
CREATE TABLE IF NOT EXISTS `PodcastEpisode` (
`episodeTitle` VARCHAR(100),
`podcastID` INT,
`duration` INT NOT NULL,
`releaseDate` DATE  NOT NULL,
`listeningCount` INT NOT NULL,
`advertisementCount` INT NOT NULL,
PRIMARY KEY(`podcastID`, `episodeTitle`),
FOREIGN KEY(`podcastID`) REFERENCES Podcast(`podcastID`) ON
DELETE CASCADE ON UPDATE CASCADE
);
```

PodcastGenre (podcastID, genreName)

```
CREATE TABLE IF NOT EXISTS `PodcastGenre` (
`podcastID` INT,
`genreName` VARCHAR(50),
PRIMARY KEY(`podcastID`, `genreName`),
FOREIGN KEY(`podcastID`) REFERENCES Podcast(`podcastID`) ON
DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(`genreName`) REFERENCES Genre(`genreName`) ON
DELETE CASCADE ON UPDATE CASCADE
);
```

PodcastSubscribers(podcastID, userAccountID, rating)

```
CREATE TABLE IF NOT EXISTS `PodcastSubscribers` (
`podcastID` INT,
`userAccountID` INT,
`rating` DECIMAL,
PRIMARY KEY(`podcastID`, `userAccountID`),
FOREIGN KEY(`podcastID`) REFERENCES Podcast(`podcastID`) ON
DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(`userAccountID`) REFERENCES
Account(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

Sponsor (sponsorID, sponsorName)

```
CREATE TABLE IF NOT EXISTS `Sponsor` (
`sponsorID` INT,
`sponsorName` VARCHAR(100) NOT NULL,
PRIMARY KEY(`sponsorID`)
);
```

AffiliatedWith (sponsorID, podcastID, amount)

```
CREATE TABLE IF NOT EXISTS `AffiliatedWith` (
`podcastID` INT,
`sponsorID` INT,
`amount` INT NOT NULL,
PRIMARY KEY(`podcastID`, `sponsorID`),
FOREIGN KEY(`podcastID`) REFERENCES Podcast(`podcastID`) ON
DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY(`sponsorID`) REFERENCES Sponsor(`sponsorID`) ON
DELETE CASCADE ON UPDATE CASCADE
);
```

UserPayment (paymentID, userAccountID, paymentDate)

```
CREATE TABLE IF NOT EXISTS `UserPayment` (
`paymentID` INT NOT NULL AUTO_INCREMENT,
`userAccountID` INT,
```

```
`paymentDate` DATE NOT NULL,
PRIMARY KEY (`paymentID`),
FOREIGN KEY (`userAccountID`) REFERENCES
`User`(`userAccountID`) ON DELETE SET NULL ON UPDATE
CASCADE
);
```

PaysLabel (paymentID, songTitle, albumID, recordLabelAccountID, paymentDate, playCount, amount)

```
CREATE TABLE IF NOT EXISTS `PaysLabel` (
`paymentID` INT NOT NULL AUTO_INCREMENT,
`recordLabelAccountID` INT NOT NULL,
`albumID` INT,
`songTitle` VARCHAR(50),
`paymentDate` DATE NOT NULL,
`playCount` INT NOT NULL,
`amount` INT NOT NULL,
PRIMARY KEY (`paymentID`),
FOREIGN KEY (`recordLabelAccountID`) REFERENCES
`RecordLabel`(`recordLabelAccountID`) ON DELETE CASCADE ON
UPDATE CASCADE,
FOREIGN KEY(`albumID`, `songTitle`) REFERENCES
`Song`(`albumID`, `songTitle`) ON DELETE SET NULL ON UPDATE
CASCADE
);
```

PaysArtist (paymentID, artistAccountID, recordLabelAccountID, songTitle, albumID, paymentDate, amount)

```
CREATE TABLE IF NOT EXISTS `PaysArtist` (
`paymentID` INT NOT NULL AUTO_INCREMENT,
`artistAccountID` INT NOT NULL,
`recordLabelAccountID` INT,
`albumID` INT,
`songTitle` VARCHAR(50),
`paymentDate` DATE NOT NULL,
`amount` INT NOT NULL,
PRIMARY KEY (`paymentID`),
FOREIGN KEY (`artistAccountID`) REFERENCES
`Artist`(`artistAccountID`) ON DELETE CASCADE ON UPDATE
CASCADE,
FOREIGN KEY (`recordLabelAccountID`) REFERENCES
`RecordLabel`(`recordLabelAccountID`) ON DELETE SET NULL ON
UPDATE CASCADE,
FOREIGN KEY(`albumID`, `songTitle`) REFERENCES
`Song`(`albumID`, `songTitle`) ON DELETE SET NULL ON UPDATE
CASCADE
);
```

PaysHost (paymentID, podcastHostAccountID, podcastID, episodeTitle, paymentDate, amount)

```
CREATE TABLE IF NOT EXISTS `PaysHost` (
`paymentID` INT NOT NULL AUTO_INCREMENT,
`podcastHostAccountID` INT,
`podcastID` INT,
`episodeTitle` VARCHAR(100),
`paymentDate` DATE NOT NULL,
`amount` INT NOT NULL,
PRIMARY KEY(`paymentID`),
FOREIGN KEY(`podcastID`, `episodeTitle`) REFERENCES
PodcastEpisode(`podcastID`, `episodeTitle`) ON DELETE SET
NULL ON UPDATE CASCADE,
FOREIGN KEY(`podcastHostAccountID`) REFERENCES
Account(`accountID`) ON DELETE CASCADE ON UPDATE CASCADE
);
```

SpecialGuest (guestID, guestName)

```
CREATE TABLE IF NOT EXISTS `SpecialGuest` (
`guestID` INT,
`guestName` VARCHAR(100) NOT NULL,
PRIMARY KEY(`guestID`)
);
```

Invites (guestID, podcastID, episodeTitle)

```
CREATE TABLE IF NOT EXISTS `Invites` (
`guestID` INT,
`podcastID` INT,
`episodeTitle` VARCHAR(50),
PRIMARY KEY(`guestID`, `podcastID`, `episodeTitle`),
FOREIGN KEY(`podcastID`, `episodeTitle`) REFERENCES
PodcastEpisode(`podcastID`, `episodeTitle`) ON DELETE
CASCADE ON UPDATE CASCADE,
FOREIGN KEY(`guestID`) REFERENCES SpecialGuest(`guestID`)
ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
SELECT * from Account;
+-----------+-------------+-----------------------+------------------+
| accountID | type        | password              | registrationDate |
+-----------+-------------+-----------------------+------------------+
|         1 | admin       | admin                 | 2023-03-01       |
|         2 | podcasthost | password              | 2023-03-02       |
|         3 | podcasthost | alohomora             | 2023-02-20       |
|         4 | podcasthost | catchmeifyoucan       | 2023-01-15       |
|         5 | podcasthost | pass                  | 2022-03-02       |
|         6 | podcasthost | test                  | 2022-07-20       |
|        12 | user        | unlock                | 2022-01-15       |
|        13 | user        | password              | 2022-05-13       |
|        14 | user        | alohomora             | 2023-02-20       |
|        15 | user        | catchmeifyoucan       | 2021-06-07       |
|        16 | artist      | Metallica             | 2012-02-12       |
|        17 | artist      | Linkin Park           | 2013-10-12       |
|        18 | artist      | Taylor Swift          | 2016-05-06       |
|        19 | artist      | Ed Sheeran            | 2012-01-01       |
|        20 | artist      | BTS                   | 2011-02-14       |
|        21 | artist      | Bruno Mars            | 2017-03-15       |
|        22 | artist      | Marshmello            | 2022-09-19       |
|        23 | artist      | Snoop Dog             | 2021-06-02       |
|        24 | artist      | Doctor Dre            | 2020-03-11       |
|        25 | artist      | Queen                 | 2015-11-16       |
|        26 | artist      | Beatles               | 2013-12-29       |
|        27 | artist      | Anderson Paak         | 2013-12-29       |
|        28 | recordLabel | Universal Music group | 2002-02-09       |
|        29 | recordLabel | Blackened recordings  | 2003-06-19       |
|        30 | recordLabel | Big Machine Records   | 2001-08-01       |
|        31 | recordLabel | Death Row Records     | 2007-05-04       |
|        32 | recordLabel | Big hit music         | 2005-02-06       |
|        33 | recordLabel | Atlantic records      | 2005-02-06       |
|        34 | user        | Batman                | 2020-10-01       |
|        35 | user        | Robin                 | 2020-07-05       |
|        36 | user        | Joker                 | 2020-09-12       |
|        37 | user        | Superman              | 2019-12-22       |
+-----------+-------------+-----------------------+------------------+
32 rows in set (0.0028 sec)

SELECT * FROM User;
+--------------+-----------+-------------+------------+------------------+----------------------+
| userAccountID | firstName | lastName    | phone      | email            | statusOfSubscription |
+--------------+-----------+-------------+------------+------------------+----------------------+
|           12 | Srilekha  | Gudipati    | 9193454359 | sngudipa@ncsu.edu | active               |
|           13 | Teja      | Manthena    | 9196787689 | smanthe@ncsu.edu  | active               |
|           14 | Varun     | Gudhe       | 4692886778 | vgudhe@ncsu.edu   | expired              |
|           15 | Sasank    | Marabattula | 9696969696 | smaraba@ncsu.edu  | expired              |
|           34 | Batman    | Fatcan      | 1234567890 | email@batman.com  | active               |
|           35 | Robin     | Obama       | 987654321  | robin@batman.com  | active               |
|           36 | Joker     | Sharma      | 9199199199 | joker@gmail.com   | active               |
|           37 | Superman  | Clark       | 147852369  | sups@gmail.com    | active               |
+--------------+-----------+-------------+------------+------------------+----------------------+
8 rows in set (0.0027 sec)
```

```
SELECT * FROM Genre;
+--------------+
| genreName    |
+--------------+
| Blues        |
| Classic Rock |
| Classical    |
| Country      |
| EDM          |
| Electronic   |
| Filmy        |
| Health       |
| HeavyMetal   |
| Hip Hop      |
| HipHop       |
| Jazz         |
| Mellow       |
| Metal        |
| Poetry       |
| Pop          |
| R&B          |
| Rap          |
| Reggae       |
| Rock         |
| SelfHelp     |
| StoryTime    |
| Tech         |
+--------------+
23 rows in set (0.0027 sec)
```

```
SELECT * FROM Artist;
+-----------------+--------------+---------+------+---------+--------------+-----------------+----------------------+
| artistAccountID | artistName   | status  | type | country | primaryGenre | monthlyListeners | recordLabelAccountID |
+-----------------+--------------+---------+------+---------+--------------+-----------------+----------------------+
|              16 | Metallica    | active  | Band | USA     | Metal        |               2 |                   29 |
|              17 | Linkin Park  | active  | Band | USA     | Rock         |               3 |                   29 |
|              18 | Taylor Swift | active  | Solo | USA     | Pop          |               0 |                   30 |
|              19 | Ed Sheeran   | active  | Solo | UK      | Pop          |               0 |                   30 |
|              20 | BTS          | active  | Band | SK      | K-Pop        |               0 |                   32 |
|              21 | Bruno Mars   | active  | Solo | USA     | R&B          |               0 |                   33 |
|              22 | Marshmello   | active  | Solo | USA     | EDM          |               0 |                   33 |
|              23 | Snoop Dog    | active  | Solo | USA     | Rap          |               0 |                   31 |
|              24 | Doctor Dre   | active  | Solo | USA     | Rap          |               0 |                   31 |
|              25 | Queen        | retired | Band | UK      | R&B          |               0 |                   28 |
|              26 | Beatles      | retired | Band | USA     | Classic Rock |               0 |                   28 |
|              27 | Anderson Paak | active | Solo | USA     | R&B          |               0 |                   30 |
+-----------------+--------------+---------+------+---------+--------------+-----------------+----------------------+
12 rows in set (0.0026 sec)
```

```
SELECT * FROM RecordLabel;
+----------------------+-----------------------+
| recordLabelAccountID | labelName             |
+----------------------+-----------------------+
|                   28 | Universal Music group |
|                   29 | Blackened recordings  |
|                   30 | Big Machine Records   |
|                   31 | Death Row Records     |
|                   32 | Big hit music         |
|                   33 | Atlantic records      |
+----------------------+-----------------------+
6 rows in set (0.0026 sec)
```

```
SELECT * FROM Album;
+---------+----------------------------+-------------+------------------+
| albumID | albumName                  | releaseYear | edition          |
+---------+----------------------------+-------------+------------------+
|       1 | Master of Puppets          |        1986 | Limited Edition  |
|       2 | Hybrid Theory              |        2000 | Special Edition  |
|       3 | Taylor Swift               |        2006 | Platinum Edition |
|       4 | X                          |        2014 | Limited Edition  |
|       5 | Still D.R.E                |        1999 | Special Edition  |
|       6 | An evening with silk Sonic |        2021 | Platinum Edition |
|       7 | Joytime                    |        2016 | Platinum Edition |
|       8 | Queen                      |        1973 | Limited Edition  |
|       9 | Abbey Road                 |        1969 | Special Edition  |
+---------+----------------------------+-------------+------------------+
9 rows in set (0.0030 sec)
```

```
SELECT * FROM Song;
+---------+--------------------+----------+-----------+-------------+----------------+----------+-------------+-------------+------------+
| albumID | songTitle          | duration | playCount | releaseDate | releaseCountry | language | royaltyRate | trackNumber | mainArtist |
+---------+--------------------+----------+-----------+-------------+----------------+----------+-------------+-------------+------------+
|       1 | Battery            | 00:05:39 |    500001 | 2013-02-25  | United States  | English  |        0.25 |           2 |         16 |
|       1 | Master of Puppets  | 00:05:20 |   1000003 | 2012-10-28  | United States  | English  |        0.25 |           1 |         16 |
|       2 | In the end         | 00:05:32 |   1000000 | 2011-07-29  | United States  | English  |        0.15 |           2 |         17 |
|       2 | Papercut           | 00:07:27 |    750000 | 2018-08-25  | United Kingdom | English  |        0.15 |           1 |         17 |
|       3 | Cold as you        | 00:05:57 |   2000000 | 2015-11-30  | United States  | English  |        0.20 |           1 |         18 |
|       3 | End game           | 00:04:18 |   1500000 | 2013-02-14  | United States  | English  |        0.20 |           2 |         18 |
|       4 | Photograph         | 00:05:01 |    900000 | 2018-09-10  | United States  | English  |        0.25 |           1 |         19 |
|       4 | Thinking out loud  | 00:03:39 |    500000 | 1992-03-02  | United States  | English  |        0.25 |           2 |         19 |
|       5 | Forgot about D.R.E. | 00:04:21 |   1000000 | 2017-01-06  | United Kingdom | English  |        0.30 |           2 |         24 |
|       5 | Still D.R.E.       | 00:03:54 |   3000000 | 2017-01-06  | United Kingdom | English  |        0.30 |           1 |         24 |
|       6 | Leave the door open | 00:03:19 |   2000000 | 2020-08-21  | South Korea    | English  |        0.25 |           2 |         21 |
|       6 | Silk Sonic intro   | 00:03:27 |    500000 | 2020-11-20  | United States  | English  |        0.25 |           1 |         21 |
|       7 | Keep it mellow     | 00:05:54 |   1500000 | 1995-10-31  | United Kingdom | English  |        0.20 |           2 |         22 |
|       7 | Show you           | 00:04:30 |    120000 | 2022-01-01  | United States  | English  |        0.05 |           1 |         22 |
|       8 | Bohemian Rhapsody  | 00:03:29 |    800000 | 1989-01-26  | United Kingdom | English  |        0.20 |           1 |         25 |
|       8 | Dont stop me now   | 00:03:45 |     12000 | 1991-11-20  | United Kingdom | English  |        0.16 |           2 |         25 |
|       9 | Dancing in the Rain | 00:05:20 |    150000 | 1998-12-31  | France         | English  |        0.08 |           2 |         26 |
|       9 | Here comes the sun | 00:04:28 |    600000 | 1987-02-04  | United Kingdom | English  |        0.30 |           1 |         26 |
+---------+--------------------+----------+-----------+-------------+----------------+----------+-------------+-------------+------------+
18 rows in set (0.0017 sec)
```

```
SELECT * FROM Releases;
+-----------------+---------+
| artistAccountID | albumID |
+-----------------+---------+
|              16 |       1 |
|              17 |       2 |
|              18 |       3 |
|              19 |       4 |
|              21 |       6 |
|              22 |       7 |
|              23 |       5 |
|              24 |       5 |
|              25 |       8 |
|              26 |       9 |
|              27 |       6 |
+-----------------+---------+
11 rows in set (0.0024 sec)
```

```
SELECT * FROM SongGenre;
+--------------+---------+---------------------+
| genreName    | albumID | songTitle           |
+--------------+---------+---------------------+
| Classic Rock |       9 | Dancing in the Rain |
| Classic Rock |       9 | Here comes the sun  |
| EDM          |       4 | Photograph          |
| EDM          |       7 | Keep it mellow      |
| EDM          |       7 | Show you            |
| Jazz         |       6 | Leave the door open |
| Jazz         |       6 | Silk Sonic intro    |
| Metal        |       1 | Battery             |
| Metal        |       1 | Master of Puppets   |
| Metal        |       2 | Papercut            |
| Pop          |       3 | Cold as you         |
| Pop          |       3 | End game            |
| Pop          |       4 | Thinking out loud   |
| Pop          |       7 | Keep it mellow      |
| R&B          |       6 | Silk Sonic intro    |
| R&B          |       8 | Bohemian Rhapsody   |
| Rap          |       3 | End game            |
| Rap          |       5 | Forgot about D.R.E. |
| Rap          |       5 | Still D.R.E.        |
| Rock         |       2 | In the end          |
| Rock         |       2 | Papercut            |
| Rock         |       8 | Bohemian Rhapsody   |
+--------------+---------+---------------------+
22 rows in set (0.0025 sec)

SELECT * FROM Collaborators;
+-----------------+---------+---------------------+
| artistAccountID | albumID | songTitle           |
+-----------------+---------+---------------------+
|              18 |       7 | Keep it mellow      |
|              19 |       3 | End Game            |
|              21 |       3 | End Game            |
|              21 |       5 | Forgot about D.R.E. |
|              22 |       4 | Photograph          |
|              26 |       8 | Dont stop me now    |
+-----------------+---------+---------------------+
6 rows in set (0.0021 sec)

SELECT * FROM ArtistSubscribers;
+-----------------+---------------+
| artistAccountID | userAccountID |
+-----------------+---------------+
|              16 |            12 |
|              16 |            13 |
|              17 |            12 |
|              17 |            13 |
|              17 |            14 |
+-----------------+---------------+
5 rows in set (0.0024 sec)
```

SELECT * FROM Podcast;

podcastID	podcastName	language	country	episodeCount	rating	totalSubscribers
1	The Minimalists	English	United Kingdom	3	4.00	1
2	The Genius Life	English	United States	5	NULL	1
3	On the Metal	English	United States	2	4.50	3
4	Short Wave	Japanese	Japan	2	NULL	1
5	On Purpose	English	United States	4	4.00	2

5 rows in set (0.0024 sec)

SELECT * FROM PodcastHost;

podcastHostAccountID	firstName	lastName	phone	email	city
2	Jay	Shetty	9192342345	jay.shetty@gmail.com	Raleigh
3	Emily	Kwong	9191232345	em.kwong@gmail.com	Okinawa
4	Bryan	Cantrill	9193334444	bcantrill@gmail.com	New York
5	Ryan	Nicodemus	9191223333	ryannic@gmail.com	London
6	Max	Lugavere	9191231234	max.lugavere@gmail.com	San Jose

5 rows in set (0.0025 sec)

SELECT * FROM PodcastEpisode;

episodeTitle	podcastID	duration	releaseDate	listeningCount	advertisementCount
Declutter	1	68	2023-02-11	1	2
Slowing Down	1	39	2023-02-21	0	3
Starting Minimalism	1	44	2023-03-01	0	2
Anti-depressant Foods	2	32	2023-03-06	0	2
Fat Loss Hacks	2	61	2023-03-07	0	2
How to Meal prep like a Boss	2	32	2023-03-05	0	1
Ketosis for A Better Brain	2	60	2023-03-07	0	2
Life Hack that Changes Your Life	2	32	2023-03-06	0	2
Oxide and Friends	3	65	2023-02-21	0	1
Star Simpson	3	96	2023-03-01	0	1
Black Gardening	4	15	2023-01-22	0	1
Disordered Cosmos	4	16	2023-02-22	0	1
Became a Monk for 3 years	5	28	2023-01-28	0	1
Lilly Singh	5	76	2023-03-01	0	3
Novak Djokovic	5	60	2023-02-25	0	3
Russell Brand	5	44	2023-02-21	0	1

16 rows in set (0.0030 sec)

SELECT * FROM Hosts;

podcastHostAccountID	podcastID
2	5
3	4
4	3
5	1
6	2

5 rows in set (0.0023 sec)

```
SELECT * FROM PodcastGenre;
+-----------+-----------+
| podcastID | genreName |
+-----------+-----------+
|         1 | SelfHelp  |
|         2 | Health    |
|         3 | Tech      |
|         4 | Poetry    |
|         5 | StoryTime |
+-----------+-----------+
5 rows in set (0.0024 sec)

SELECT * FROM PodcastSubscribers;
+-----------+---------------+--------+
| podcastID | userAccountID | rating |
+-----------+---------------+--------+
|         1 |            12 |      4 |
|         2 |            12 |   NULL |
|         3 |            12 |   NULL |
|         3 |            13 |      4 |
|         3 |            14 |      5 |
|         4 |            13 |   NULL |
|         5 |            12 |      4 |
|         5 |            14 |   NULL |
+-----------+---------------+--------+
8 rows in set (0.0025 sec)


SELECT * FROM Sponsor;
+-----------+---------------+
| sponsorID | sponsorName   |
+-----------+---------------+
|       300 | Bill Gates    |
|       301 | Elon Musk     |
|       302 | Barack Obama  |
|       303 | Rada Chirkova |
+-----------+---------------+
4 rows in set (0.0024 sec)

SELECT * FROM AffiliatedWith ;
+-----------+-----------+--------+
| podcastID | sponsorID | amount |
+-----------+-----------+--------+
|         1 |       302 |   4000 |
|         1 |       303 |  15000 |
|         3 |       302 |   5000 |
|         4 |       300 |  12000 |
+-----------+-----------+--------+
4 rows in set (0.0025 sec)

SELECT * FROM UserPayment;
+-----------+---------------+-------------+
| paymentID | userAccountID | paymentDate |
+-----------+---------------+-------------+
|         8 |            12 | 2023-03-11  |
|         9 |            13 | 2023-03-11  |
|        10 |            34 | 2023-03-11  |
```

```
|         11 |              35 | 2023-03-11  |
|         12 |              36 | 2023-03-11  |
|         13 |              37 | 2023-03-11  |
+------------+-----------------+-------------+
6 rows in set (0.0023 sec)
```

```
SELECT * FROM PaysLabel;
```

paymentID	recordLabelAccountID	albumID	songTitle	paymentDate	playCount	amount
1	28	8	Bohemian Rhapsody	2023-03-11	800000	160000
2	28	8	Dont stop me now	2023-03-11	12000	1920
3	28	9	Dancing in the Rain	2023-03-11	150000	12000
4	28	9	Here comes the sun	2023-03-11	600000	180000
5	29	1	Battery	2023-03-11	500001	125000
6	29	1	Master of Puppets	2023-03-11	1000003	250001
7	29	2	In the end	2023-03-11	1000000	150000
8	29	2	Papercut	2023-03-11	750000	112500
9	30	3	Cold as you	2023-03-11	2000000	400000
10	30	3	End game	2023-03-11	1500000	300000
11	30	4	Photograph	2023-03-11	900000	225000
12	30	4	Thinking out loud	2023-03-11	500000	125000
13	31	5	Forgot about D.R.E.	2023-03-11	1000000	300000
14	31	5	Still D.R.E.	2023-03-11	3000000	900000
15	33	6	Leave the door open	2023-03-11	2000000	500000
16	33	6	Silk Sonic intro	2023-03-11	500000	125000
17	33	7	Keep it mellow	2023-03-11	1500000	300000
18	33	7	Show you	2023-03-11	120000	6000
38	29	1	Master of Puppets	2023-02-28	500000	125000
39	28	8	Dont stop me now	2023-02-28	24000	3840
40	30	4	Photograph	2023-02-28	1800000	450000
42	29	8	Dont stop me now	2023-01-31	250000	62500
43	31	5	Forgot about D.R.E.	2023-02-28	2000000	600000
44	31	5	Forgot about D.R.E.	2023-01-31	1000000	300000

24 rows in set (0.0027 sec)

```
SELECT * FROM PaysArtist;
```

paymentID	artistAccountID	recordLabelAccountID	albumID	songTitle	paymentDate	amount
33	25	28	8	Bohemian Rhapsody	2023-03-12	112000
34	25	28	8	Dont stop me now	2023-03-12	672
35	26	28	8	Dont stop me now	2023-03-12	672
36	26	28	9	Dancing in the Rain	2023-03-12	8400
37	26	28	9	Here comes the sun	2023-03-12	126000
38	16	29	1	Battery	2023-03-12	87500
39	16	29	1	Master of Puppets	2023-03-12	175001
40	17	29	2	In the end	2023-03-12	105000
41	17	29	2	Papercut	2023-03-12	78750
42	18	30	3	Cold as you	2023-03-12	280000
43	18	30	3	End game	2023-03-12	69993
44	21	30	3	End game	2023-03-12	69993
45	19	30	3	End game	2023-03-12	69993
46	19	30	4	Photograph	2023-03-12	78750
47	22	30	4	Photograph	2023-03-12	78750
48	19	30	4	Thinking out loud	2023-03-12	87500
49	24	31	5	Forgot about D.R.E.	2023-03-12	69993
50	23	31	5	Forgot about D.R.E.	2023-03-12	69993
51	21	31	5	Forgot about D.R.E.	2023-03-12	69993
52	24	31	5	Still D.R.E.	2023-03-12	315000
53	23	31	5	Still D.R.E.	2023-03-12	315000
54	27	33	6	Leave the door open	2023-03-12	175000
55	21	33	6	Leave the door open	2023-03-12	175000
56	27	33	6	Silk Sonic intro	2023-03-12	43750
57	21	33	6	Silk Sonic intro	2023-03-12	43750
58	22	33	7	Keep it mellow	2023-03-12	105000
59	18	33	7	Keep it mellow	2023-03-12	105000
60	22	33	7	Show you	2023-03-12	4200

28 rows in set (0.0027 sec)

```
SELECT * FROM PaysHost;
```

paymentID	podcastHostAccountID	podcastID	episodeTitle	paymentDate	amount
63	2	5	Became a Monk for 3 years	2023-03-12	350
64	2	5	Lilly Singh	2023-03-12	550
65	2	5	Novak Djokovic	2023-03-12	550
66	2	5	Russell Brand	2023-03-12	350
67	3	4	Black Gardening	2023-03-12	350
68	3	4	Disordered Cosmos	2023-03-12	350
69	4	3	Oxide and Friends	2023-03-12	350
70	4	3	Star Simpson	2023-03-12	350
71	5	1	Declutter	2023-03-12	450
72	5	1	Slowing Down	2023-03-12	550
73	5	1	Starting Minimalism	2023-03-12	450
74	6	2	Anti-depressant Foods	2023-03-12	450
75	6	2	Fat Loss Hacks	2023-03-12	450
76	6	2	How to Meal prep like a Boss	2023-03-12	350
77	6	2	Ketosis for A Better Brain	2023-03-12	450
78	6	2	Life Hack that Changes Your Life	2023-03-12	450

16 rows in set (0.0014 sec)

```
SELECT * FROM SpecialGuest;
```

guestID	guestName
700	Ian Somelhalder
701	Emma Watson
702	Shah Rukh Khan
703	Jessica Alba
704	Tom Holland
705	Candice king

6 rows in set (0.0032 sec)

```
SELECT * FROM Invites;
```

guestID	podcastID	episodeTitle
700	1	Declutter
701	5	Became a Monk for 3 years
702	3	Star Simpson
703	2	Fat Loss Hacks
704	4	Disordered Cosmos
705	2	Anti-depressant Foods

6 rows in set (0.0023 sec)

IV. SQL Queries

4.1. Tasks and Operations

1. **Information Processing**

Enter/update/delete basic information about songs.

```
INSERT INTO Song (`albumID`, `songTitle`, `duration`,
`playCount`, `releaseDate`, `releaseCountry`, `language`,
`royaltyRate`, `trackNumber`, `mainArtist`)
VALUES
(1, 'Master of Puppets', '0:5:20', 1000000, '2012-10-28',
'United States', 'English', 0.25, 1, 16);
```

```
Query OK, 1 row affected (0.00 sec)
```

```
UPDATE Song SET `duration` = '0:6:31'
WHERE `songTitle` = 'Master of Puppets' AND `albumID` = 1;
```

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
DELETE FROM Song
WHERE  `songTitle` = 'Show you' and `albumID` = 7;
```

```
Query OK, 1 row affected (0.01 sec)
```

Enter/update/delete basic information about artists.

```
INSERT INTO Artist (`artistAccountID`, `artistName`,
`status`,`type`,`country`,`primaryGenre`,
`monthlyListeners`, `recordLabelAccountID`) VALUES (4,
'Enrique Iglesias', 'musician', 'USA', 'pop', 50000,
'Republic records');
```

```
Query OK, 1 row affected (0.00 sec)
```

```
UPDATE Artist SET `country`='United States' WHERE
artistAccountID=4;
```

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

```
DELETE FROM Artist where artistAccountID=4;
```

```
Query OK, 1 row affected (0.01 sec)
```

Enter/update/delete basic information about podcast hosts.

```
INSERT INTO PodcastHost (`podcastHostAccountID`,
`firstName`, `lastName`, `phone`, `email`, `city`) VALUES
(6, 'Maxx', 'Lugavere', '9191231234',
'max.lugavere@gmail.com' , 'San Jose');
```

Query OK, 1 row affected (0.00 sec)

```
UPDATE PodcastHost SET firstName = 'Max'  WHERE
podcastHostAccountID=6;
```

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```
DELETE FROM PodcastHost where podcastHostAccountID=6;
```

Query OK, 1 row affected (0.01 sec)

Enter/update/delete basic information about podcast episodes.

```
INSERT INTO PodcastEpisode (`podcastID`, `episodeTitle`,
`duration`, `releaseDate`, `listeningCount`,
`advertisementCount`) VALUES (1, 'How to Meal prep like a
Boss', 32, "2023-03-05", 500 , 1);
```

Query OK, 1 row affected (0.00 sec)

```
UPDATE PodcastEpisode SET advertisementCount=2 WHERE
podcastID=1 AND episodeTitle='How to Meal prep like a
Boss';
```

Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

```
DELETE FROM PodcastEpisode where podcastID=1 AND
episodeTitle='Declutter';
```

Query OK, 1 row affected (0.01 sec)

Assign songs and artists to albums.

Assign songs to album:

```
INSERT INTO Song (`albumID`, `songTitle`, `duration`,
`playCount`, `releaseDate`, `releaseCountry`, `language`,
`royaltyRate`, `trackNumber`, `mainArtist`)
VALUES
(1, 'Master of Puppets', '0:5:20', 1000000, '2012-10-28',
'United States', 'English', 0.25, 1, 16);
```

Query OK, 1 row affected (0.00 sec)

Assign Artist to album:

```
Insert into Releases(artistAccountID, albumID) values (16,
1);
```

Query OK, 1 row affected (0.00 sec)

Assign artists to record labels.

```
Insert into Artist (artistAccountID, artistName, status,
type, country, primaryGenre, monthlyListeners,
recordLabelAccountID)
values
(18,'Taylor Swift', 'active', 'Solo', 'USA', 'Pop', 0, 30);
```

Query OK, 1 row affected (0.00 sec)

Assign podcast episodes.

```
INSERT INTO PodcastEpisode (`podcastID`, `episodeTitle`,
`duration`, `releaseDate`, `listeningCount`,
`advertisementCount`)
VALUES(`1`, `How to Meal prep like a Boss`, 32, "2023-03-
05", 500 , 1);
```

Query OK, 1 row affected (0.00 sec)

Assign podcast hosts to podcasts.

```
INSERT INTO Podcast (`podcastID`, `podcastName`,
`language`, `country`, `episodeCount`, `rating`,
`totalSubscribers`)
VALUES (2, `The Genius Life`, `English`, `United States`,
398, NULL, NULL);
```

Query OK, 1 row affected (0.00 sec)

```
INSERT INTO Hosts (`podcasHostAccouttID`,` podcastID`)
VALUES (`6`, `2`);
```

Query OK, 1 row affected (0.00 sec)

2. **Maintaining metadata and records**

Enter/update play count for songs.

```
UPDATE Song SET playCount = playCount+1 WHERE albumID=1 AND
songTitle LIKE "Master of Puppets";
```

Query OK, 1 row affected (0.0584 sec)
Rows matched: 1 Changed: 1 Warnings: 0

Enter/update the count of monthly listeners for artists.

```
UPDATE Artist a1 SET monthlyListeners=(SELECT count(*) FROM
ArtistSubscribers a2 WHERE
a1.artistAccountID=a2.artistAccountID);
```

```
Query OK, 9 row affected (0.0584 sec)
Rows matched: 12  Changed: 9  Warnings: 0
```

Enter/update the total count of subscribers and ratings for podcasts.

```
UPDATE Podcast p1 SET totalSubscribers=(SELECT count(*)
FROM PodcastSubscribers p2 WHERE
p1.podcastID=p2.podcastID);
```

```
Query OK, 5 row affected (0.031 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

```
UPDATE Podcast p1 SET rating=(SELECT AVG(rating) FROM
PodcastSubscribers p2 WHERE p1.podcastID=p2.podcastID);
```

```
Query OK, 5 row affected (0.021 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

Enter/Update the listening count for podcast episodes.

```
UPDATE PodcastEpisode SET listeningCount = listeningCount +
1 WHERE podcastID=1 AND episodeTitle='Declutter';
```

```
Query OK, 16 row affected (0.0578 sec)
Rows matched: 16  Changed: 16  Warnings: 0
```

Find songs given artistID

```
SELECT s.albumID, s.songTitle
FROM Song s NATURAL JOIN Collaborators c
WHERE c.artistAccountID=18
UNION SELECT s.albumID, s.songTitle
FROM Song s NATURAL JOIN Releases r NATURAL JOIN Album a
Where r.artistAccountID=18;
```

```
+---------+----------------+
| albumID | songTitle      |
+---------+----------------+
|       7 | Keep it mellow |
|       3 | Cold as you    |
|       3 | End game       |
+---------+----------------+
3 rows in set (0.0112 sec)
```

```
SELECT c.artistAccountID, s.albumID, s.songTitle
FROM Song s NATURAL JOIN Collaborators c
UNION
```

```
SELECT r.artistAccountID, s.albumID, s.songTitle
FROM Song s NATURAL JOIN Releases r NATURAL JOIN Album a;
```

```
+-----------------+---------+---------------------+
| artistAccountID | albumID | songTitle           |
+-----------------+---------+---------------------+
|              18 |       7 | Keep it mellow      |
|              19 |       3 | End game            |
|              21 |       3 | End game            |
|              21 |       5 | Forgot about D.R.E. |
|              22 |       4 | Photograph          |
|              26 |       8 | Dont stop me now    |
|              16 |       1 | Battery             |
|              16 |       1 | Master of Puppets   |
|              17 |       2 | In the end          |
|              17 |       2 | Papercut            |
|              18 |       3 | Cold as you         |
|              18 |       3 | End game            |
|              19 |       4 | Photograph          |
|              19 |       4 | Thinking out loud   |
|              23 |       5 | Forgot about D.R.E. |
|              23 |       5 | Still D.R.E.        |
|              24 |       5 | Forgot about D.R.E. |
|              24 |       5 | Still D.R.E.        |
|              21 |       6 | Leave the door open |
|              21 |       6 | Silk Sonic intro    |
|              27 |       6 | Leave the door open |
|              27 |       6 | Silk Sonic intro    |
|              22 |       7 | Keep it mellow      |
|              22 |       7 | Show you            |
|              25 |       8 | Bohemian Rhapsody   |
|              25 |       8 | Dont stop me now    |
|              26 |       9 | Dancing in the Rain |
|              26 |       9 | Here comes the sun  |
+-----------------+---------+---------------------+
28 rows in set (0.0018 sec)
```

Find songs given albumID

```
SELECT s.albumID, s.songTitle
FROM Song s
Where s.albumID=5;
```

```
+---------+---------------------+
| albumID | songTitle           |
+---------+---------------------+
|       5 | Forgot about D.R.E. |
|       5 | Still D.R.E.        |
+---------+---------------------+
2 rows in set (0.0018 sec)
```

Find Podcast episode given podcastID

```
Select episodeTitle, podcastID, duration, releaseDate
FROM PodcastEpisode
WHERE podcastID=4;
```

```
+------------------+-----------+----------+-------------+
| episodeTitle     | podcastID | duration | releaseDate |
+------------------+-----------+----------+-------------+
| Black Gardening  |         4 |       15 | 2023-01-22  |
| Disordered Cosmos |         4 |       16 | 2023-02-22  |
+------------------+-----------+----------+-------------+
2 rows in set (0.0017 sec)
```

Find Podcast episode given podcastHostAccountID

```
Select episodeTitle, podcastID, duration, releaseDate
FROM Hosts NATURAL JOIN  PodcastEpisode
WHERE podcastHostAccountID=2;
```

```
+--------------------------+-----------+----------+-------------+
| episodeTitle             | podcastID | duration | releaseDate |
+--------------------------+-----------+----------+-------------+
| Became a Monk for 3 years |         5 |       28 | 2023-01-28  |
| Lilly Singh              |         5 |       76 | 2023-03-01  |
| Novak Djokovic           |         5 |       60 | 2023-02-25  |
| Russell Brand            |         5 |       44 | 2023-02-21  |
+--------------------------+-----------+----------+-------------+
4 rows in set (0.0018 sec)
```

3. Maintaining payments

Make royalty payments for a given song.

```
INSERT INTO PaysLabel (songTitle, albumID,
recordLabelAccountID, paymentDate, playCount, amount)
SELECT s.songTitle, s.albumID, r.recordLabelAccountID,
CURDATE(), s.playCount, (s.playCount * s.royaltyRate)
FROM
Song s, RecordLabel r, Artist a WHERE s.mainArtist =
a.ArtistAccountID AND a.recordLabelAccountID =
r.recordLabelAccountID AND s.songTitle='Battery';
```

```
Query OK, 1 row affected (0.0034 sec)
Records: 1  Duplicates: 0  Warnings: 0
```

Make monthly royalty payment for all songs to the record labels:

```
INSERT INTO PaysLabel (songTitle, albumID,
recordLabelAccountID, paymentDate, playCount, amount)
SELECT s.songTitle, s.albumID, r.recordLabelAccountID,
CURDATE(), s.playCount, (s.playCount * s.royaltyRate)
```

```
FROM
Song s, RecordLabel r, Artist a WHERE s.mainArtist =
a.ArtistAccountID AND a.recordLabelAccountID =
r.recordLabelAccountID;
```

Query OK, 18 row affected (0.0048 sec)
Records: 18 Duplicates: 0 Warnings: 0

Make monthly royalty payment for all songs to the artists

```
CREATE VIEW SongsPerArtist AS
SELECT a1.recordLabelAccountID, c.artistAccountID,
s.albumID, s.songTitle, MONTH(CURDATE()) as Month,
YEAR(CURDATE()) as Year, playCount, (royaltyRate *
playCount) AS amount
FROM Song s NATURAL JOIN Collaborators c
INNER JOIN Artist a1 ON a1.artistAccountID=s.mainArtist
UNION
SELECT a2.recordLabelAccountID, r.artistAccountID,
s.albumID, s.songTitle, MONTH(CURDATE()) as Month,
YEAR(CURDATE()) as Year, playCount, (royaltyRate *
playCount) AS amount
FROM Song s NATURAL JOIN Releases r NATURAL JOIN Album a
INNER JOIN Artist a2 ON a2.artistAccountID=s.mainArtist;
```

Query OK, 0 rows affected (0.0100 sec)

```
INSERT INTO `PaysArtist` (artistAccountID,
recordLabelAccountID, albumID, songTitle, paymentDate,
amount)
SELECT z.artistAccountID, y.recordLabelAccountID,
y.albumID, y.songTitle, CURDATE(), (0.7 * y.percent *
z.amount)
FROM (SELECT x.recordLabelAccountID, x.albumID,
x.songTitle, (1/count(*)) AS percent
FROM SongsPerArtist x
GROUP BY x.recordLabelAccountID, x.albumID, x.songTitle) y,
SongsPerArtist z
WHERE y.albumID=z.albumID AND y.songTitle=z.songTitle;
```

Query OK, 28 row affected (0.0084 sec)
Records: 28 Duplicates: 0 Warnings: 0

Make payment to podcast hosts.

```
INSERT INTO PaysHost (`podcastHostAccountID`, `podcastID`,
`episodeTitle`, `paymentDate`, `amount`)
```

```
SELECT Hosts.podcastHostAccountID, Hosts.podcastID,
PodcastEpisode.episodeTitle,CURDATE(),(250 +
PodcastEpisode.advertisementCount*100)
FROM
Hosts,PodcastEpisode WHERE
Hosts.podcastID=PodcastEpisode.podcastID
AND NOT EXISTS
(SELECT `podcastHostAccountID`, `podcastID`,
`episodeTitle`, `paymentDate`, `amount` FROM PaysHost);
```

```
Query OK, 16 row affected (0.0040 sec)
Records: 16  Duplicates: 0  Warnings: 0
```

Receive payment from subscribers.

```
INSERT INTO UserPayment(userAccountID, paymentDate)
SELECT userAccountID, CURDATE() FROM User where
statusOfSubscription = `active`;
```

```
Query OK, 6 row affected (0.0043 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

4. **Reports**

Monthly play count per song.

```
SELECT albumID, songTitle, playCount, MONTH(CURDATE()) as
Month, YEAR(CURDATE()) as Year FROM Song
UNION
SELECT albumID, songTitle, playCount, MONTH(paymentDate) as
Month, YEAR(paymentDate) as Year FROM PaysLabel;
```

albumID	songTitle	playCount	Month	Year
1	Battery	500001	3	2023
1	Master of Puppets	1000004	3	2023
2	In the end	1000000	3	2023
2	Papercut	750000	3	2023
3	Cold as you	2000000	3	2023
3	End game	1500000	3	2023
4	Photograph	900000	3	2023
4	Thinking out loud	500000	3	2023
5	Forgot about D.R.E.	1000000	3	2023
5	Still D.R.E.	3000000	3	2023
6	Leave the door open	2000000	3	2023
6	Silk Sonic intro	500000	3	2023
7	Keep it mellow	1500000	3	2023
7	Show you	120000	3	2023
8	Bohemian Rhapsody	800000	3	2023
8	Dont stop me now	12000	3	2023
9	Dancing in the Rain	150000	3	2023

```
|       9 | Here comes the sun  |    600000 |     3 | 2023 |
|       1 | Master of Puppets   |    500000 |     2 | 2023 |
|       8 | Dont stop me now    |     24000 |     2 | 2023 |
|       4 | Photograph          |   1800000 |     2 | 2023 |
|       8 | Dont stop me now    |    250000 |     1 | 2023 |
|       5 | Forgot about D.R.E. |   2000000 |     2 | 2023 |
|       5 | Forgot about D.R.E. |   1000000 |     1 | 2023 |
+---------+---------------------+-----------+-------+------+
24 rows in set (0.0016 sec)
```

Monthly play count per album

```
SELECT albumID, songTitle, SUM(playCount) as playCount,
MONTH(CURDATE()) as Month, YEAR(CURDATE()) as Year FROM
Song GROUP by albumID
UNION
SELECT albumID, songTitle, SUM(playCount) as playCount,
MONTH(paymentDate) as Month, YEAR(paymentDate) as Year FROM
PaysLabel Group by albumID;
```

```
+---------+---------------------+-----------+-------+------+
| albumID | songTitle           | playCount | Month | Year |
+---------+---------------------+-----------+-------+------+
|       1 | Battery             |   1500005 |     3 | 2023 |
|       2 | In the end          |   1750000 |     3 | 2023 |
|       3 | Cold as you         |   3500000 |     3 | 2023 |
|       4 | Photograph          |   1400000 |     3 | 2023 |
|       5 | Forgot about D.R.E. |   4000000 |     3 | 2023 |
|       6 | Leave the door open |   2500000 |     3 | 2023 |
|       7 | Keep it mellow      |   1620000 |     3 | 2023 |
|       8 | Bohemian Rhapsody   |    812000 |     3 | 2023 |
|       9 | Dancing in the Rain |    750000 |     3 | 2023 |
|       1 | Battery             |   2000005 |     3 | 2023 |
|       4 | Photograph          |   3200000 |     3 | 2023 |
|       5 | Forgot about D.R.E. |   7000000 |     3 | 2023 |
|       8 | Bohemian Rhapsody   |   1086000 |     3 | 2023 |
+---------+---------------------+-----------+-------+------+
13 rows in set (0.0018 sec)
```

Monthly play count per artist

```
CREATE VIEW PlayCountPerSong AS
SELECT c.artistAccountID, s.albumID, s.songTitle,
MONTH(CURDATE()) as Month, YEAR(CURDATE()) as Year,
playCount
FROM Song s NATURAL JOIN Collaborators c
UNION SELECT r.artistAccountID,  s.albumID, s.songTitle,
MONTH(CURDATE()) as Month, YEAR(CURDATE()) as Year,
playCount
FROM Song s NATURAL JOIN Releases r NATURAL JOIN Album a;
```

```
Query OK, 0 rows affected (0.0100 sec)
```

```
CREATE VIEW PlayCountPerPaysLabel AS
SELECT c.artistAccountID, s.albumID, s.songTitle,
MONTH(s.paymentDate) as Month, YEAR(s.paymentDate) as Year,
playCount
FROM PaysLabel s NATURAL JOIN Collaborators c
UNION SELECT r.artistAccountID,  s.albumID, s.songTitle,
MONTH(s.paymentDate) as Month, YEAR(s.paymentDate) as Year,
playCount
FROM PaysLabel s NATURAL JOIN Releases r NATURAL JOIN Album
a;
```

Query OK, 0 rows affected (0.0120 sec)

```
SELECT  artistAccountID, Month, Year, SUM(playCount)
FROM (SELECT * FROM PlayCountPerSong
UNION
SELECT * FROM PlayCountPerPaysLabel) x
GROUP BY artistAccountID, Month, Year;
```

artistAccountID	Month	Year	SUM(playCount)
16	2	2023	500000
16	3	2023	1500005
17	3	2023	1750000
18	3	2023	5000000
19	2	2023	1800000
19	3	2023	2900000
21	1	2023	1000000
21	2	2023	2000000
21	3	2023	5000000
22	2	2023	1800000
22	3	2023	2520000
23	1	2023	1000000
23	2	2023	2000000
23	3	2023	4000000
24	1	2023	1000000
24	2	2023	2000000
24	3	2023	4000000
25	1	2023	250000
25	2	2023	24000
25	3	2023	812000
26	1	2023	250000
26	2	2023	24000
26	3	2023	762000
27	3	2023	2500000

24 rows in set (0.0043 sec)

Calculate total payments made out to Record Label per a given time period.

```
Select recordLabelAccountID, SUM(amount) as totalPayment
FROM PaysLabel
WHERE paymentDate>='2023-01-01' AND paymentDate<=CURDATE()
```

```
            GROUP BY recordLabelAccountID;
```

```
+----------------------+--------------+
| recordLabelAccountID | totalPayment |
+----------------------+--------------+
|                   28 |       357760 |
|                   29 |       825001 |
|                   30 |      1500000 |
|                   31 |      2100000 |
|                   33 |       931000 |
+----------------------+--------------+
5 rows in set (0.0018 sec)
```

Calculate total payments made out to Artist per a given time period.

```
            Select artistAccountID, SUM(amount) as totalPayment
            FROM PaysArtist
            WHERE paymentDate>='2023-01-01' AND paymentDate<=CURDATE()
            GROUP BY artistAccountID;
```

```
+-----------------+--------------+
| artistAccountID | totalPayment |
+-----------------+--------------+
|              16 |       262501 |
|              17 |       183750 |
|              18 |       454993 |
|              19 |       236243 |
|              21 |       358736 |
|              22 |       187950 |
|              23 |       384993 |
|              24 |       384993 |
|              25 |       112672 |
|              26 |       135072 |
|              27 |       218750 |
+-----------------+--------------+
11 rows in set (0.0021 sec)
```

Calculate total payments made out to host per a given time period.

```
            SELECT podcastHostAccountID, SUM(amount) as totalPayment
            FROM PaysHost
            WHERE paymentDate>='2023-01-01' AND paymentDate<=CURDATE()
            GROUP BY podcastHostAccountID;
```

```
+----------------------+--------------+
| podcastHostAccountID | totalPayment |
+----------------------+--------------+
|                    2 |         1800 |
|                    3 |          700 |
|                    4 |          700 |
|                    5 |         1450 |
|                    6 |         2150 |
+----------------------+--------------+
5 rows in set (0.0019 sec)
```

Total revenue of the streaming service per month.

```
SELECT MONTH(paymentDate), YEAR(paymentDate), (COUNT(*) *
20) AS `Total revenue`
FROM UserPayment
GROUP BY MONTH(paymentDate), YEAR(paymentDate);
```

```
+--------------------+-------------------+---------------+
| MONTH(paymentDate) | YEAR(paymentDate) | Total revenue |
+--------------------+-------------------+---------------+
|                  1 |              2023 |            60 |
|                  2 |              2023 |           120 |
|                  3 |              2023 |           120 |
|                 12 |              2022 |            60 |
+--------------------+-------------------+---------------+
4 rows in set (0.0018 sec)
```

Total revenue of the streaming service per year

```
SELECT YEAR(paymentDate), (COUNT(*) * 20) AS `Total
revenue`
FROM UserPayment
GROUP BY YEAR(paymentDate);
```

```
+-------------------+---------------+
| YEAR(paymentDate) | Total revenue |
+-------------------+---------------+
|              2022 |            60 |
|              2023 |           300 |
+-------------------+---------------+
2 rows in set (0.0017 sec)
```

Report all songs given an artist

```
SELECT s.albumID, s.songTitle, s.playCount
FROM Song s NATURAL JOIN Collaborators c
WHERE c.artistAccountID=18
UNION SELECT s.albumID, s.songTitle, s.playCount
FROM Song s NATURAL JOIN Releases r NATURAL JOIN Album a
Where r.artistAccountID=18;
```

```
+---------+----------------+-----------+
| albumID | songTitle      | playCount |
+---------+----------------+-----------+
|       7 | Keep it mellow |   1500000 |
|       3 | Cold as you    |   2000000 |
|       3 | End game       |   1500000 |
+---------+----------------+-----------+
3 rows in set (0.0023 sec)
```

Report all songs given albumID

```
SELECT * FROM Song
Where albumID=5;
```

```
+---------+--------------------+----------+-----------+-------------+----------------+----------+-------------+-------------+------------+
| albumID | songTitle          | duration | playCount | releaseDate | releaseCountry | language | royaltyRate | trackNumber | mainArtist |
+---------+--------------------+----------+-----------+-------------+----------------+----------+-------------+-------------+------------+
|       5 | Forgot about D.R.E. | 00:04:21 |   1000000 | 2017-01-06  | United Kingdom | English  |        0.30 |           2 |         24 |
|       5 | Still D.R.E.       | 00:03:54 |   3000000 | 2017-01-06  | United Kingdom | English  |        0.30 |           1 |         24 |
+---------+--------------------+----------+-----------+-------------+----------------+----------+-------------+-------------+------------+
2 rows in set (0.0017 sec)
```

Report all Podcast episode given podcastID

```
Select *
FROM PodcastEpisode
WHERE podcastID=5;
```

```
+--------------------------+-----------+----------+-------------+----------------+---------------------+
| episodeTitle             | podcastID | duration | releaseDate | listeningCount | advertisementCount |
+--------------------------+-----------+----------+-------------+----------------+---------------------+
| Became a Monk for 3 years |         5 |       28 | 2023-01-28  |              0 |                   1 |
| Lilly Singh              |         5 |       76 | 2023-03-01  |              0 |                   3 |
| Novak Djokovic           |         5 |       60 | 2023-02-25  |              0 |                   3 |
| Russell Brand            |         5 |       44 | 2023-02-21  |              0 |                   1 |
+--------------------------+-----------+----------+-------------+----------------+---------------------+
4 rows in set (0.0017 sec)
```

Report all Podcast episode given podcastHostAccountID

```
Select *
FROM Hosts NATURAL JOIN  PodcastEpisode
WHERE podcastHostAccountID=2;
```

```
+-----------+----------------------+--------------------------+----------+-------------+----------------+---------------------+
| podcastID | podcastHostAccountID | episodeTitle             | duration | releaseDate | listeningCount | advertisementCount |
+-----------+----------------------+--------------------------+----------+-------------+----------------+---------------------+
|         5 |                    2 | Became a Monk for 3 years |       28 | 2023-01-28  |              0 |                   1 |
|         5 |                    2 | Lilly Singh              |       76 | 2023-03-01  |              0 |                   3 |
|         5 |                    2 | Novak Djokovic           |       60 | 2023-02-25  |              0 |                   3 |
|         5 |                    2 | Russell Brand            |       44 | 2023-02-21  |              0 |                   1 |
+-----------+----------------------+--------------------------+----------+-------------+----------------+---------------------+
4 rows in set (0.0017 sec)
```

Report all Podcast episode given releaseDate:

```
select * from PodcastEpisode WHERE releaseDate='2023-02-25';
+----------------+-----------+----------+-------------+----------------+---------------------+
| episodeTitle   | podcastID | duration | releaseDate | listeningCount | advertisementCount |
+----------------+-----------+----------+-------------+----------------+---------------------+
| Novak Djokovic |         5 |       60 | 2023-02-25  |              0 |                   3 |
+----------------+-----------+----------+-------------+----------------+---------------------+
1 row in set (0.0062 sec)
```

4.2. Analyzing SQL Queries using EXPLAIN directive and optimizing using Indexes:

i. Return podcast episode by release date:

```
explain select * from PodcastEpisode WHERE releaseDate='2023-02-25';
```

```
+----+-------------+----------------+------+---------------+------+---------+------+------+-------------+
| id | select_type | table          | type | possible_keys | key  | key_len | ref  | rows | Extra       |
+----+-------------+----------------+------+---------------+------+---------+------+------+-------------+
|  1 | SIMPLE      | PodcastEpisode | ALL  | NULL          | NULL | NULL    | NULL |   16 | Using where |
+----+-------------+----------------+------+---------------+------+---------+------+------+-------------+
1 row in set (0.0024 sec)
```

Here, where clause is being used in the query, and any rows with releaseDate='2023-02-25' are being searched for by performing a full table scan of the PodcastEpisode table.

This can be improved by avoiding a full table scan by adding an index to the releaseDate.

```
create index podcastReleaseIndex on PodcastEpisode(releaseDate);
```

```
Query OK, 0 rows affected (0.0402 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
explain select * from PodcastEpisode WHERE releaseDate='2023-02-25';
```

```
+----+-------------+----------------+------+---------------------+---------------------+---------+-------+------+-----------------------+
| id | select_type | table          | type | possible_keys       | key                 | key_len | ref   | rows | Extra                 |
+----+-------------+----------------+------+---------------------+---------------------+---------+-------+------+-----------------------+
|  1 | SIMPLE      | PodcastEpisode | ref  | podcastReleaseIndex | podcastReleaseIndex | 3       | const |    1 | Using index condition |
+----+-------------+----------------+------+---------------------+---------------------+---------+-------+------+-----------------------+
1 row in set (0.0024 sec)
```

Now that the index has been added to the table, it is clear that we are utilizing it to find the entries in the table.

ii. Calculate total payments made out to Record Label per a given time period.

```
Select recordLabelAccountID, SUM(amount) as totalPayment
FROM PaysLabel
WHERE paymentDate>='2023-01-01' AND paymentDate<=CURDATE()
GROUP BY recordLabelAccountID;
```

```
+----+-------------+-----------+-------+---------------+----------------------+---------+------+------+-------------+
| id | select_type | table     | type  | possible_keys | key                  | key_len | ref  | rows | Extra       |
+----+-------------+-----------+-------+---------------+----------------------+---------+------+------+-------------+
|  1 | SIMPLE      | PaysLabel | index | NULL          | recordLabelAccountID | 4       | NULL |   27 | Using where |
+----+-------------+-----------+-------+---------------+----------------------+---------+------+------+-------------+
1 row in set (0.0025 sec)
```

```
create index PaysLabelPeriod on PaysLabel(paymentDate);
```

```
Query OK, 0 rows affected (0.0392 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
Select recordLabelAccountID, SUM(amount) as totalPayment
FROM PaysLabel
```

```
WHERE paymentDate>='2023-01-01' AND paymentDate<=CURDATE()
GROUP BY recordLabelAccountID;
```

```
+----+-------------+-----------+-------+----------------+---------------------+---------+------+------+-----------------------+
| id | select_type | table     | type  | possible_keys  | key                 | key_len | ref  | rows | Extra                 |
+----+-------------+-----------+-------+----------------+---------------------+---------+------+------+-----------------------+
|  1 | SIMPLE      | PaysLabel | index | PaysLabelPeriod | recordLabelAccountID | 4       | NULL |   27 | Using index condition |
+----+-------------+-----------+-------+----------------+---------------------+---------+------+------+-----------------------+
1 row in set (0.0026 sec)
```

4.3. Query Correctness:

Calculate total payments made out to Artist per a given time period:

```
Select artistAccountID, SUM(amount) as totalPayment
FROM PaysArtist
WHERE paymentDate>='2023-01-01' AND paymentDate<=CURDATE()
GROUP BY artistAccountID;
```

$$\gamma_{artistAccountID,SUM(amount)}(\sigma_{paymentDate\geq'2023-01-01'ANDpaymentDate\leq'2023-03-12'}(paysArtist))$$

Consider a time period that begins on January 1st, 2023, and ends on the present day. Any account ID in the PaysArtist relation that appears in at least one tuple and meets the requirement that the payment date falls within the specified time frame shall be referred to as artistAccountID. The query calculates the total of all payments made to that artist during the specified time frame and names it totalPayment.

The query's WHERE clause makes sure that only payments made during the specified time period are taken into account, and the GROUP BY clause makes sure that the total is calculated differently for each artist. The query thus returns the account ID of the artist and the total amount of payments made to that artist for each artist who received payments during the specified time period. The query specification specifically calls for this. In light of this, we can say that the query answer always matches the query specification.

Find songs given artistID:

```
SELECT s.albumID, s.songTitle
FROM Song s NATURAL JOIN Collaborators c
WHERE c.artistAccountID=18
UNION SELECT s.albumID, s.songTitle
FROM Song s NATURAL JOIN Releases r NATURAL JOIN Album a
Where r.artistAccountID=18;
```

$$\pi_{s.albumID,s.songTitle}(\sigma_{c.artistAccountID=18}(\rho_s(Song)\bowtie\rho_c(Collaborators)))\cup \pi_{s.albumID,s.songTitle}(\sigma_{r.artistAccountID=18}(\rho_s(Song)\bowtie\rho_r(Releases)\bowtie \rho_a(Album)))$$

Take Song and Collaborators as our hypothetical two relations. The Song relation contains details about songs, such as the song title and album ID. Information about song and artist collaborations, including the artist ID and album ID, is contained in the collaborators relation. A third relation in our database, called Releases, contains details about album releases, such as the artist ID and album ID. Utilizing the UNION operator, the query gathers data from both relations.

Finding all songs with a specific artist ID (in this case, artistAccountID = 18) is the query's specification. In order to achieve this, the query performs a natural join between the Song and Collaborators relations on the album ID, and then filters the resulting tuples to only contain those with artistAccountID = 18. This provides us with all songs connected to that artist via collaborations. The query then executes a second natural join on the album ID between the Song, Releases, and Album relations and filters the resulting tuples to only contain those with artistAccountID = 18. This provides us with all the songs connected to that artist's album releases. The two sets of results are finally combined into one set of tuples by the UNION operator.

We can reason as follows to show that the query answer always matches the query specification: for any artist with ID 18, the query will return all songs connected to that artist through collaborations and album releases. Due to the query's natural joins between the Song and Collaborators relations and the Song, Releases, and Album relations, both on the album ID, these results are obtained. These procedures make sure that only the songs connected to the specified artist through band or album releases are retrieved. The query specification demands that the returned set of tuples include the album ID and song title for each song connected to the artist.

Since the query returns all songs connected to the specified artist ID, we can infer that the query answer always matches the query specification.