WolfMedia Database Management System

Team S

Ameya Girish Vaichalkar (agvaicha)

Kartik Hemendra Rawool (khrawool)

Dhrumil Jignesh Shah (dshah6)

Subodh Suryakant Gujar (sgujar)

Contents

Assumptions

1. Artist can have a contract with exactly one record label at a time.
2. Podcast is deleted then all the episodes within that podcast will also be deleted.
3. Album is deleted then all the album tracks within that album will also be deleted.
4. All artists, record labels, and podcast hosts are paid on first day of the month.
5. Users pays subscription fees on the first day of the month.
6. Monthly active listeners include only the users who have subscribed to the artist or the podcast host.
7. Phone numbers entered by users are unique.
8. Email-ID entered by users are unique.
9. Song will always be part of at most one album.
10. Song will be sung in only one language.
11. Each track number is unique within an album. For eg., multiple albums can have same track number.
12. Each episode number is unique within a podcast. For eg., multiple podcast episode can have the same episode number.
13. Different tables are maintained for keeping the records of song history, artist history, and podcast history.
14. Data will be added in History tables only at the last day of the month.
15. Each song has exactly one primary artist.
16. Royalties will be paid for the songs to the record label of the primary artist.
17. Service account represents all the transactions done by and to WolfMedia Streaming service.
18. Podcast ratings will be in the range of 0-5.
19. There are multiple plans for subscription to WolfMedia. Each plan has different subscription fees.
20. All key attributes are NON NULL.
21. All referential integrity will have ON UPDATE CASCADE and ON DELETE CASCADE constraint.

1 Global Relational Database Schema

1. **song(<u>songId</u>, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId)**

   - Functional Dependencies:
     (a) songId → songId, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId
   - Above FD holds because each songId is unique, and uniquely identifies a song that has a royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId. If we were to try and take any combination of other attributes, it would severely limit the possibilities our database could have. For example, many songs have the same primaryArtist, language and country. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

2. **album(<u>albumId</u>, name, releaseYear, edition)**

   - Functional Dependencies:
     (a) albumId → albumId, name, releaseYear, edition
   - This FD holds because each albumId is unique, and uniquely identifies an album along with its name, year of release, and edition. Two or more albums can have the same name, edition, and release year; hence these attributes cannot be part of a superkey. Because albumId is a superkey, this functional dependency is in BCNF (and thereby 3NF).

3. **trackNumber(<u>albumId</u>, <u>number</u>)**

   - Functional Dependencies:
     (a) albumId, number → albumId, number
   - This is a trivial FD that holds because there are only two attributes, and are both are a part of the super key, so it is in BCNF and therefore, 3NF.

4. **genre(<u>genreId</u>, name)**

   - Functional Dependencies:
     (a) genreId → genreId, name
   - This FD holds because a genreId is used to uniquely identify a genre. Because the left hand side is a super key, it is in BCNF (and thereby 3NF).

5. **SongHas(<u>songId</u>,<u>genreId</u>)**

   - Functional Dependencies:
     (a) songId, genreId → songId, genreId
   - This is a trivial FD that holds because there are only two attributes, and are both are a part of the super key, so it is in BCNF and therefore, 3NF.

6. **artistHas(<u>artistId</u>, <u>albumId</u>)**
   - Functional Dependencies:
     (a) artistId, albumId → artistId, albumId
   - This is a trivial FD that holds because there are only two attributes, and are both are a part of the super key, so it is in BCNF and therefore, 3NF.
7. **creates(<u>artistId</u>, <u>songId</u>, type)**
   - Functional Dependencies:
     (a) artistId, songId → artistId, songId, type
   - holds because a artistId, songId is used to uniquely identify the type of artist(guest artist, collaborated artist); the type of the artist cannot be a superkey as it does not determine other attributes. Because the left-hand side is a super key, it is in BCNF (and thereby 3NF).
8. **Podcast(<u>podcastId</u>, name, country, language, rating, episodeCount, totalSubscribers)**
   - Functional Dependancies:
     (a) podcastId → podcastId, name, country, language, rating, episodeCount, totalSubscribers
   - podcastId can uniquely identify all other attributes of the podcast. Podcast name and country can be the same for two or more podcasts. Other combinations of attributes cannot uniquely identify a podcast. The above relation is in BCNF and, therefore, 3NF because the left-hand side of the functional dependency is a superkey.
9. **episode(<u>podcastId</u>, <u>number</u>, title, duration, releaseDate, ListeningCount, AdvertisementCount)**
   - Functional Dependancies:
     (a) podcastId, number → podcastId, number, title, duration, releaseDate, ListeningCount, AdvertisementCount
   - podcastId alone cannot uniquely identify all other attributes of the podcast episode because a podcast can have multiple episodes. PodcastId, along with episode number, can uniquely identify podcast title, duration, releaseDate, listening count, and AdvertisementCount. Other combinations of attributes example, number, and title, cannot uniquely identify a podcast, as episode number or a title can be the same for episodes that belong to different podcasts. The above relation is in BCNF and, therefore, 3NF because the left-hand side of the functional dependency is a superkey.
10. **episodeFeaturesGuest(<u>podcastId</u>, <u>number</u>, <u>guestId</u>)**
    - Functional Dependencies:
      (a) podcastId, number, guestId → podcastId, number, guestId

- podcastId, number, guestId → podcastId, number, guestId holds because the relationship "features" is connected to two entities, one which is a weak entity set. This relationship requires the keys for all entities it is connected to, which would be the three we have listed. Without any of the three listed in the key, we can not appropriately describe the relationship; for instance, if the guestId is not listed then we cannot identify who the guestId is associated with. Because the left hand side contains all attributes, it is in BCNF (and thereby 3NF).

11. **podcastHas(<u>podcastId</u>, <u>genreId</u>)**

- Functional Dependencies:
  (a) podcastId, genreId → podcastId, genreId
- podcastId, genreId → podcastId, genreId is in BCNF and therefore 3NF because it only contains two attributes, which are in the superkey so it is in BCNF and therefore 3NF.

12. **podcastHistory(<u>podcastId</u>, month, year, subscribers, rating)**

- Functional Dependencies:
  (a) podcastId → podcastId, month, year, subscribers, rating
- podcastId → podcastId, month, year, subscribers, rating is true since each podcast has a unique id (podcastId) that determines the month, year, subscribers, and rating of the podcast.
- This relation is in BCNF and therefore 3NF because the left hand side of the only functional dependency is a superkey.

13. **specialGuests(<u>guestId</u>, name)**

- Functional Dependencies:
  (a) guestId → guestId, name
- guestId → guestId, name holds since this relation only has two attributes, it is in BCNF and therefore 3NF. name cannot be used as a key because multiple guests can have the same name, but it cannot indicate a guests unique guestId.

14. **createdBy(<u>podcastId</u>, <u>hostId</u>)**

- Functional Dependencies:
  (a) podcastId, hostId → podcastId, hostId
- podcastId, hostId → podcastId, hostId is in BCNF and therefore 3NF because it only contains two attributes, which are in the superkey so it is in BCNF and therefore 3NF.

15. **sponsors(<u>sponsorId</u>, name, amount)**

- Functional Dependencies:

(a) sponsorId → sponsorId, name, amount

- sponsorId → sponsorId, name, amount is true since each sponsor has a unique id (sponsorId) that determines the month, year, subscribers, and rating of the podcast.
- This relation is in BCNF and therefore 3NF because the left hand side of the only functional dependency is a superkey.

16. **sponseredBy(<u>sponsorId</u>, <u>podcastId</u>)**

- Functional Dependencies:

(a) sponsorId, podcastId → podcastId, sponsorId

- sponsorId, podcastId → podcastId, sponsorId is in BCNF and therefore 3NF because it only contains two attributes, which are in the superkey so it is in BCNF and therefore 3NF.

17. **artist(<u>artistId</u>, name, status, country, primaryGenre, monthlyListeners, recordId)**

- Functional Dependencies:

(a) artistId → artistId, name, status, country, primaryGenre, monthlyListeners, recordId

- Above FD holds because each artistId is unique, and uniquely identifies a album that has a name, status, country, primaryGenre, monthlyListeners, recordId. If we were to try and take any combination of other attributes, it would severely limit the possibilities our database could have. For example, many artist have the same recordId, country and primaryGenre. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

18. **belongsTo(<u>artistTypeId</u>, <u>artistId</u>)**

- Functional Dependencies:

(a) artistTypeId, artistId → artistTypeId, artistId

- Above FD holds because there are only two attributes and are both in the super key, so it is in BCNF and therefore 3NF.

19. **artistType(<u>artistTypeId</u>, type)**

- Functional Dependencies:

(a) artistTypeId → artistTypeId, type

- Above FD holds because there are only two attributes and are both in the super key, so it is in BCNF and therefore 3NF.

20. **collaboratedWith(<u>artistId1</u>, <u>artistId2</u>)**

- Functional Dependencies:

(a) artistId1, artistId2 → artistId1, artistId2

- Above FD holds because there are only two attributes and are both in the super key, so it is in BCNF and therefore 3NF.

21. **recordLabel(<u>recordId</u>, name)**

- Functional Dependencies:
  (a) recordId → recordId, name
- Above FD holds because there are only two attributes and are both in the super key, so it is in BCNF and therefore 3NF.

22. **receives(<u>transactionId</u>, <u>recordId</u>)**

- Functional Dependencies:
  (a) transactionId, recordId → transactionId, recordId
- Above FD holds because there are only two attributes and are both in the super key, so it is in BCNF and therefore 3NF.

23. **user(<u>userId</u>, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate)**

- Functional Dependencies:
  (a) userId → userId, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate
  (b) phone → userId, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate
  (c) email → userId, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate
- FD (a) holds because each user with all of their respective attributes has a unique userID. FD (b) holds because each user with all of their respective attributes has a unique phone based on assumption. FD (c) holds because each user with all of their respective attributes has a unique email. Because the left hand side of each of these functional dependencies is a superkey, this relation is in BCNF and therefore 3NF. There are no other functional dependencies because no combination of the other attributes is sufficient to determine a unique user. It's possible for two different user to have identical values for each of the remaining attributes.

24. **serviceAccount(<u>transactionId</u>, date, amount, type)**

- Functional Dependencies:
  (a) transactionId → transactionId, date, amount, type
- Above FD holds because each transactionId is unique, and uniquely identifies a transaction that has a date, amount, type. If we were to try and take any combination of other attributes, it would severely limit the possibilities our database could have. For example, many transaction have the same date and type. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

25. **songHistory(<u>songId</u>, <u>month</u>, <u>year</u>, playCount)**

    - Functional Dependencies:
      (a) songId, month, year → songId, month, year, playCount
    - Above FD holds since each combination of songId, month and year is unique, and therefore uniquely identifies a songHistory that has a playCount for every month. Closure of above FD results into all attributes of schema. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

26. **pays(<u>userId</u>, <u>transactionId</u>)**

    - Functional Dependencies:
      (a) userId, transactionId → userId, transactionId
    - Above FD holds because its trivial FD. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

27. **podcastHost(<u>hostId</u>, firstName, lastName, contact, email, flatFee, city)**

    - Functional Dependancies:
      (a) hostId → hostId, firstName, lastName, contact, email, flatFee, city
      (b) contact → hostId, firstName, lastName, contact, email, flatFee, city
      (c) email → hostId, firstName, lastName, contact, email, flatFee, city
    - Above FDs holds because each podcastId or contact or email is unique, and uniquely identifies a podcast that has a podcastId, firstName, lastName, contact, email, flatFee, city. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

28. **follow(<u>artistId</u>, <u>userId</u>)**

    - Functional Dependencies:
      (a) artistId, userId → artistId, userId
    - Above FD holds because its trivial functional dependency. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

29. **subscribes(<u>podcastId</u>, <u>userId</u>)**

    - Functional Dependencies:
      (a) podcastId, userId → podcastId, userId
    - Above FD holds because its trivial functional dependency. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

30. **givesPaymentTo(<u>transactionId</u>, <u>HostId</u>)**

    - Functional Dependencies:

(a) transactionId, HostId → transactionId, HostId

- Above FD holds because its trivial functional dependency. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

31. **artistHistory(<u>artistId</u>, <u>Month</u>, <u>Year</u>, Subscribers)**

- Functional Dependencies:

  (a) artistId, month, year → artistId, month, year, Subscribers

- Above FD holds since each combination of artistId, month and year is unique, and therefore uniquely identifies a artistHistory tuple that has a Subscribers for every month. Closure of above FD results into all attributes of schema. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

32. **distributesroyalties(<u>artistId</u>, <u>recordId</u>, songId, <u>date</u>, amount)**

- Functional Dependencies:

  (a) artistId, recordId, date → artistId, recordId, date, songId, amount

- Above FD holds since each combination of artistId, recordId and date is unique, and therefore uniquely identifies a artistHistory tuple that has a artistId, recordId, date, songId, amount for every month. Closure of above FD results into all attributes of schema. Because the left hand side is a superkey, this functional dependency is in BCNF (and thereby 3NF).

2 Design Decisions for the Global Schema

1. **song(songId, royaltyRate, title, royaltyStatus, playCount, country, language, duration, primaryArtist, albumId)**
   - Keys
     (a) songId
   - NULL Attributes
     (a) country
     (b) language
     (c) When entering the song information about country and language, we might don't know actual values, therefore we will keep them as NULL and update it later.
   - NON-NULL
     (a) songId
     (b) royaltyRate
     (c) title
     (d) royaltyStatus
     (e) playCount (default value will be 0)
     (f) duration
     (g) primaryArtist
   - Referential Integrity
     (a) albumId
     (b) primaryArtist
2. **album(albumId, name, releaseYear, edition)**
   - Keys
     (a) albumId
   - NULL Attributes
     (a) edition
     (b) edition attribute can be null if album doesn't have any specific edition at the time of insertion.
   - NON-NULL
     (a) name
     (b) releaseYear
3. **trackNumber(albumId, number)**
   - Keys
     (a) albumId, number
   - Referential Integrity

   (a) albumId

4. **genre(genreId, name)**
   - Keys
     (a) genreId
   - NON-NULL
     (a) name

5. **SongHas(songId,genreId)**
   - Keys
     (a) songId, genreId
   - Referential Integrity
     (a) songId
     (b) genreId

6. **artistHas(artistId, albumId)**
   - Keys
     (a) artistId, albumId
   - Referential Integrity
     (a) artistId
     (b) albumId

7. **creates(artistId, songId, type)**
   - Keys
     (a) artistId, songId
   - NON-NULL
     (a) type
   - Referential Integrity
     (a) artistId
     (b) songId

8. **Podcast(podcastId, name, country, language, rating, episodeCount, totalSubscribers)**
   - Keys
     (a) podcastId
   - NULL Attributes
     (a) country
     (b) language

(c) When entering the podcast information about country and language, we might don't know actual values, therefore we will keep them as NULL and update it later.

- NON-NULL
  (a) name
  (b) rating (default value will be 0)
  (c) episodeCount (default value will be 0)
  (d) totalSubscribers (default value will be 0)

9. **episode(<u>podcastId</u>, <u>number</u>, title, duration, releaseDate, ListeningCount, AdvertisementCount)**

   - Keys
     (a) podcastId, number
   - NON-NULL
     (a) title
     (b) duration
     (c) releaseDate
     (d) ListeningCount (default value will be 0)
     (e) AdvertisementCount (default value will be 0)
   - Referential Integrity
     (a) podcastId

10. **episodeFeaturesGuest(<u>podcastId</u>, <u>number</u>, <u>guestId</u>)**

    - Keys
      (a) podcastId
      (b) number
      (c) guestId
    - Referential Integrity
      (a) podcastId, number
      (b) guestId

11. **podcastHas(<u>podcastId</u>, <u>genreId</u>)**

    - Keys
      (a) podcastId, genreId
    - Referential Integrity
      (a) podcastId
      (b) genreId

12. **podcastHistory(<u>podcastId</u>, month, year, subscribers, rating)**

    - Keys

    (a) podcastId
- NON-NULL
    (a) rating
    (b) subscribers
    (c) month
    (d) year

13. **specialGuests(guestId, name)**
    - Keys
        (a) guestId
    - NON-NULL
        (a) name
14. **createdBy(podcastId, hostId)**
    - Keys
        (a) podcastId, hostId
    - Referential Integrity
        (a) podcastId
        (b) hostId
15. **sponsors(sponsorId, name, amount)**
    - Keys
        (a) sponsorId
    - NON-NULL
        (a) name
        (b) amount
16. **sponseredBy(sponsorId, podcastId)**
    - Keys
        (a) sponsorId, podcastId
    - Referential Integrity
        (a) podcastId
        (b) sponsorId
17. **artist(artistId, name, status, country, primaryGenre, monthlyListeners, recordId)**
    - Keys
        (a) artistId
    - NULL Attributes

(a) country
(b) When entering information about country, we might don't know origin country of artist, therefore we will keep it as NULL and update it later.

- NON-NULL
  (a) artistId
  (b) name
  (c) status
  (d) primaryGenre
  (e) monthlyListeners
- Referential Integrity
  (a) recordId

18. **belongsTo(<u>artistTypeId</u>, <u>artistId</u>)**
    - Keys
      (a) artistTypeId, artistId
    - Referential Integrity
      (a) artistTypeId
      (b) artistId
19. **artistType(<u>artistTypeId</u>, type)**
    - Keys
      (a) artistTypeId
    - NON-NULL
      (a) type
20. **collaboratedWith(<u>artistId1</u>, <u>artistId2</u>)**
    - Keys
      (a) artistId1
      (b) artistId2
    - Referential Integrity
      (a) artistId1
      (b) artistId2
21. **recordLabel(<u>recordId</u>, name)**
    - Keys
      (a) recordId
    - NON-NULL
      (a) name

22. **receives(<u>transactionId</u>, <u>recordId</u>)**
    - Keys
      (a) transactionId
      (b) recordId
    - Referential Integrity
      (a) transactionId
      (b) recordId
23. **user(<u>userId</u>, firstName, lastName, email, subscriptionStatus, subscriptionFee, phone, registrationDate)**
    - Keys
      (a) userId
    - NON-NULL
      (a) userId
      (b) firstName
      (c) lastName
      (d) email
      (e) subscriptionStatus
      (f) subscriptionFee
      (g) phone
      (h) registrationDate
24. **serviceAccount(<u>transactionId</u>, date, amount, type)**
    - Keys
      (a) transactionId
    - NON-NULL
      (a) date
      (b) amount
      (c) type
25. **songHistory(<u>songId</u>, <u>month</u>, <u>year</u>, playCount)**
    - Keys
      (a) songId, month, year
    - NON-NULL
      (a) songId
      (b) month
      (c) year
      (d) playCount (default 0)
    - Referential Integrity

(a) songId

26. **pays(<u>userId</u>, <u>transactionId</u>)**
    - Keys
      (a) userId, transactionId
    - Referential Integrity
      (a) userId
      (b) transactionId
27. **podcastHost(<u>hostId</u>, firstName, lastName, contact, email, flatFee, city)**
    - Keys
      (a) hostId
    - NON-NULL
      (a) hostId
      (b) contact
      (c) email
      (d) contact
      (e) email
      (f) firstName
      (g) lastName
      (h) flatFee
      (i) city
28. **follow(<u>artistId</u>, <u>userId</u>)**
    - Keys
      (a) artistId, userId
    - Referential Integrity
      (a) artistId
      (b) userId
29. **subscribes(<u>podcastId</u>, <u>userId</u>)**
    - Keys
      (a) podcastId, userId
    - Referential Integrity
      (a) podcastId
      (b) userId
30. **givesPaymentTo(<u>transactionId</u>, <u>HostId</u>)**
    - Keys

   (a) transactionId, HostId
- Referential Integrity
   (a) transactionId
   (b) HostId

31. **artistHistory(<u>artistId</u>, <u>Month</u>, <u>Year</u>, Subscribers)**
- Keys
   (a) artistId, Month, Year
- NON-NULL
   (a) artistId
   (b) Month
   (c) Year
   (d) Subscribers (default 0)
- Referential Integrity
   (a) artistId

32. **distributesroyalties(<u>artistId</u>, <u>recordId</u>, <u>songId</u>, <u>date</u>, amount)**
- Keys
   (a) artistId, recordId, date
- NON-NULL
   (a) artistId
   (b) recordId
   (c) date
   (d) amount (default 0)
   (e) songId
- Referential Integrity
   (a) artistId
   (b) recordId
   (c) songId

3 Base Relations

3.1 Create Schema Queries with constraints

```
CREATE TABLE recordLabels (
  recordId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE albums (
  albumId INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  releaseYear INT NOT NULL,
  edition VARCHAR(255)
);


CREATE TABLE artists (
    artistId INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    status VARCHAR(255) NOT NULL,
    country VARCHAR(255),
    primaryGenre VARCHAR(255) NOT NULL,
    monthlyListeners INT NOT NULL,
    recordId INT,
    PRIMARY KEY (artistId),
    FOREIGN KEY (recordId) REFERENCES recordLabels(recordId)
    ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE songs (
  songId INT PRIMARY KEY AUTO_INCREMENT,
  royaltyRate FLOAT NOT NULL,
  title VARCHAR(255) NOT NULL,
  royaltyStatus VARCHAR(255) NOT NULL,
  playCount INT NOT NULL DEFAULT 0,
  country VARCHAR(255),
  language VARCHAR(255),
  duration FLOAT NOT NULL,
  primaryArtist INT NOT NULL,
  albumId INT,
  FOREIGN KEY (albumId) REFERENCES albums(albumId)
  ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (primaryArtist) REFERENCES artists(artistId)
  ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE trackNumbers (
  albumId INT,
  number INT,
  PRIMARY KEY (albumId, number),
  FOREIGN KEY (albumId) REFERENCES albums(albumId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE genres (
  genreId INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL
);

CREATE TABLE songHas (
  songId INT,
  genreId INT,
  PRIMARY KEY (songId, genreId),
  FOREIGN KEY (songId) REFERENCES songs(songId)
  ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (genreId) REFERENCES genres(genreId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE artistHas (
  artistId INT,
  albumId INT,
  PRIMARY KEY (artistId, albumId),
  FOREIGN KEY (artistId) REFERENCES artists(artistId)
  ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (albumId) REFERENCES albums(albumId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE creates (
  artistId INT,
  songId INT,
  type VARCHAR(255) NOT NULL,
  PRIMARY KEY (artistId, songId),
  FOREIGN KEY (artistId) REFERENCES artists(artistId)
  ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (songId) REFERENCES songs(songId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE users (
  userId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  firstName VARCHAR(255) NOT NULL,
```

```
  lastName VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  subscriptionStatus VARCHAR(255) NOT NULL,
  subscriptionFee FLOAT NOT NULL,
  phone VARCHAR(20) NOT NULL UNIQUE,
  registrationDate DATE NOT NULL
);

CREATE TABLE podcasts (
  podcastId INT PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  country VARCHAR(255),
  language VARCHAR(255),
  rating FLOAT NOT NULL DEFAULT 0,
  episodeCount INT NOT NULL DEFAULT 0,
  totalSubscribers INT NOT NULL DEFAULT 0
);

CREATE TABLE episodes (
  podcastId INT,
  number INT,
  title VARCHAR(255) NOT NULL,
  duration FLOAT NOT NULL,
  releaseDate DATE NOT NULL,
  ListeningCount INT NOT NULL DEFAULT 0,
  AdvertisementCount INT NOT NULL DEFAULT 0,
  PRIMARY KEY (podcastId, number),
  FOREIGN KEY (podcastId) REFERENCES podcasts(podcastId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE specialGuests (
    guestId INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    PRIMARY KEY (guestId)
);

CREATE TABLE episodeFeaturesGuest (
  podcastId INT,
  number INT,
  guestId INT,
  PRIMARY KEY (podcastId, number, guestId),
  FOREIGN KEY (podcastId, number) REFERENCES
  episodes(podcastId, number) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (guestId) REFERENCES specialGuests(guestId)
  ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
CREATE TABLE podcastHosts (
  hostId INT NOT NULL AUTO_INCREMENT,
  firstName VARCHAR(255) NOT NULL,
  lastName VARCHAR(255) NOT NULL,
  contact VARCHAR(20) NOT NULL UNIQUE,
  email VARCHAR(255) NOT NULL UNIQUE,
  flatFee DECIMAL(10,2) NOT NULL,
  city VARCHAR(255) NOT NULL,
  PRIMARY KEY (hostId)
);

CREATE TABLE podcastHas (
    podcastId INT NOT NULL,
    genreId INT NOT NULL,
    PRIMARY KEY (podcastId, genreId),
    FOREIGN KEY (podcastId) REFERENCES
    podcasts(podcastId) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (genreId) REFERENCES
    genres(genreId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE podcastHistory (
    podcastId INT NOT NULL,
    month INT NOT NULL,
    year INT NOT NULL,
    subscribers INT NOT NULL DEFAULT 0,
    rating FLOAT NOT NULL DEFAULT 0,
    PRIMARY KEY (podcastId, month, year),
    FOREIGN KEY (podcastId) REFERENCES
    podcasts(podcastId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE createdBy (
    podcastId INT NOT NULL,
    hostId INT NOT NULL,
    PRIMARY KEY (podcastId, hostId),
    FOREIGN KEY (podcastId) REFERENCES
    podcasts(podcastId) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (hostId) REFERENCES
    podcastHosts(hostId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE sponsors (
    sponsorId INT NOT NULL AUTO_INCREMENT,
    name VARCHAR(255) NOT NULL,
    amount DECIMAL(10,2) NOT NULL,
```

```
    PRIMARY KEY (sponsorId)
);

CREATE TABLE sponsoredBy (
    sponsorId INT NOT NULL,
    podcastId INT NOT NULL,
    PRIMARY KEY (sponsorId, podcastId),
    FOREIGN KEY (sponsorId) REFERENCES
    sponsors(sponsorId) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (podcastId) REFERENCES
    podcasts(podcastId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE artistType (
    artistTypeId INT NOT NULL AUTO_INCREMENT,
    type VARCHAR(255) NOT NULL,
    PRIMARY KEY (artistTypeId)
);

CREATE TABLE belongsTo (
    artistTypeId INT NOT NULL,
    artistId INT NOT NULL,
    PRIMARY KEY (artistTypeId, artistId),
    FOREIGN KEY (artistTypeId) REFERENCES
    artistType(artistTypeId) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (artistId) REFERENCES
    artists(artistId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE collaboratedWith (
  artistId1 INT NOT NULL,
  artistId2 INT NOT NULL,
  PRIMARY KEY (artistId1, artistId2),
  FOREIGN KEY (artistId1) REFERENCES artists(artistId)
  ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (artistId2) REFERENCES artists(artistId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE serviceAccount (
  transactionId INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
  date DATE NOT NULL,
  amount FLOAT NOT NULL,
  type VARCHAR(255) NOT NULL
);

CREATE TABLE receives (
```

```
  transactionId INT NOT NULL,
  recordId INT NOT NULL,
  PRIMARY KEY (transactionId, recordId),
  FOREIGN KEY (transactionId) REFERENCES
  serviceAccount(transactionId) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (recordId) REFERENCES recordLabels(recordId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE songHistory (
  songId INT NOT NULL,
  month INT NOT NULL,
  year INT NOT NULL,
  playCount INT DEFAULT 0,
  PRIMARY KEY (songId, month, year),
  FOREIGN KEY (songId) REFERENCES songs(songId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE pays (
  userId INT NOT NULL,
  transactionId INT NOT NULL,
  PRIMARY KEY (userId, transactionId),
  FOREIGN KEY (transactionId) REFERENCES
  serviceAccount(transactionId) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (userId) REFERENCES users(userId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE follow (
  artistId INT NOT NULL,
  userId INT NOT NULL,
  PRIMARY KEY (artistId, userId),
  FOREIGN KEY (artistId) REFERENCES artists(artistId)
  ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (userId) REFERENCES users(userId)
  ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE subscribes (
  podcastId INT NOT NULL,
  userId INT NOT NULL,
  PRIMARY KEY (podcastId, userId),
  FOREIGN KEY (podcastId) REFERENCES
  podcasts(podcastId) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (userId) REFERENCES users(userId)
  ON UPDATE CASCADE ON DELETE CASCADE
```

```
);

CREATE TABLE givesPaymentTo (
  transactionId INT NOT NULL,
  hostId INT NOT NULL,
  PRIMARY KEY (transactionId, hostId),
  FOREIGN KEY (transactionId) REFERENCES
  serviceAccount(transactionId) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (hostId) REFERENCES
  podcastHosts(hostId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE artistHistory (
  artistId INT NOT NULL,
  month INT NOT NULL,
  year INT NOT NULL,
  subscribers INT DEFAULT 0,
  PRIMARY KEY (artistId, month, year),
  FOREIGN KEY (artistId) REFERENCES
  artists(artistId) ON UPDATE CASCADE ON DELETE CASCADE
);

CREATE TABLE distributesRoyalties (
  artistId INT NOT NULL,
  recordId INT NOT NULL,
  songId INT NOT NULL,
  date DATE NOT NULL,
  amount DECIMAL(10, 2) DEFAULT 0,
  PRIMARY KEY (artistId, recordId, date),
  FOREIGN KEY (artistId) REFERENCES
  artists(artistId) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (recordId) REFERENCES
  recordLabels(recordId) ON UPDATE CASCADE ON DELETE CASCADE,
  FOREIGN KEY (songId) REFERENCES
  songs(songId) ON UPDATE CASCADE ON DELETE CASCADE
);
```

3.2 Select * Queries

```
SELECT * FROM distributesRoyalties;
```

```
+----------+----------+--------+------------+--------+
| artistId | recordId | songId | date       | amount |
+----------+----------+--------+------------+--------+
|        1 |        1 |      1 | 2020-01-01 | 100.00 |
|        2 |        2 |      2 | 2020-01-01 | 200.00 |
|        3 |        3 |      3 | 2020-01-01 | 300.00 |
|        4 |        4 |      4 | 2020-01-01 | 400.00 |
|        5 |        5 |      5 | 2020-01-01 | 500.00 |
|        6 |        6 |      6 | 2020-01-01 | 600.00 |
|        7 |        7 |      7 | 2020-01-01 | 700.00 |
+----------+----------+--------+------------+--------+
7 rows in set (0.0123 sec)
```

```
SELECT * FROM artistHistory;
```

```
+----------+-------+------+-------------+
| artistId | month | year | subscribers |
+----------+-------+------+-------------+
|        1 |     1 | 2020 |         100 |
|        2 |     1 | 2020 |         200 |
|        3 |     1 | 2020 |         300 |
|        4 |     1 | 2020 |         400 |
|        5 |     1 | 2020 |         500 |
|        6 |     1 | 2020 |         600 |
|        7 |     1 | 2020 |         700 |
+----------+-------+------+-------------+
7 rows in set (0.0104 sec)
```

```
SELECT * FROM givesPaymentTo;
```

```
+---------------+--------+
| transactionId | hostId |
+---------------+--------+
|             1 |      1 |
|             2 |      2 |
|             3 |      3 |
|             4 |      4 |
|             5 |      3 |
|             6 |      2 |
|             7 |      1 |
+---------------+--------+
7 rows in set (0.0107 sec)
```

```
SELECT * FROM subscribes;
```

```
+-----------+--------+
| podcastId | userId |
+-----------+--------+
|         1 |      1 |
|         2 |      2 |
|         3 |      3 |
|         4 |      4 |
|         5 |      5 |
|         6 |      6 |
|         7 |      7 |
+-----------+--------+
7 rows in set (0.0108 sec)
```

```
SELECT * FROM follow;
```

```
+----------+--------+
| artistId | userId |
+----------+--------+
|        1 |      1 |
|        2 |      2 |
|        3 |      3 |
|        4 |      4 |
|        5 |      5 |
|        6 |      6 |
|        7 |      7 |
+----------+--------+
7 rows in set (0.0108 sec)
```

```
SELECT * FROM podcastHosts;
```

```
+--------+-----------+----------+------------+-----------------------+---------+----------------+
| hostId | firstName | lastName | contact    | email                 | flatFee | city           |
+--------+-----------+----------+------------+-----------------------+---------+----------------+
|      1 | Jhon      | Doe      | 1234567890 | jhon.doe@gmail.com    | 1000.00 | New York       |
|      2 | James     | Smith    | 1234567890 | james.smith@gmail.com | 1000.00 | New Jersey     |
|      3 | John      | Smith    | 1234567890 | jhon.smith@gmail.com  | 1000.00 | North Carolina |
|      4 | Bob       | Doe      | 1234567890 | bob.doe@gmail.com     | 1000.00 | California     |
|      5 | James     | Thanos   | 1234567890 | james.thanos@gmail.com | 1000.00 | New York       |
|      6 | Alice     | cold     | 1234567890 | alice.cold@gmail.com  | 1000.00 | Rochester      |
+--------+-----------+----------+------------+-----------------------+---------+----------------+
6 rows in set (0.0110 sec)
```

```
SELECT * FROM pays;
```

```
+--------+---------------+
| userId | transactionId |
+--------+---------------+
|      1 |             1 |
|      2 |             2 |
|      3 |             3 |
|      4 |             4 |
|      5 |             5 |
|      6 |             6 |
|      7 |             7 |
+--------+---------------+
7 rows in set (0.0106 sec)
```

```
SELECT * FROM songHistory;
```

```
+--------+-------+------+-----------+
| songId | month | year | playCount |
+--------+-------+------+-----------+
|      1 |     1 | 2020 |       100 |
|      2 |     1 | 2020 |       200 |
|      3 |     1 | 2020 |       300 |
|      4 |     1 | 2020 |       400 |
|      5 |     1 | 2020 |       500 |
|      6 |     1 | 2020 |       600 |
|      7 |     1 | 2020 |       700 |
|      8 |     1 | 2020 |       800 |
|      9 |     1 | 2020 |       900 |
|     10 |     1 | 2020 |      1000 |
+--------+-------+------+-----------+
10 rows in set (0.0090 sec)
```

```
SELECT * FROM serviceAccount;
```

```
+---------------+------------+--------+--------+
| transactionId | date       | amount | type   |
+---------------+------------+--------+--------+
|             1 | 2020-01-01 |   1000 | Credit |
|             2 | 2020-01-01 |   2000 | Credit |
|             3 | 2020-01-01 |    500 | Debit  |
|             4 | 2020-01-01 |   1000 | Credit |
|             5 | 2020-01-01 |    200 | Credit |
|             6 | 2020-01-01 |    100 | Debit  |
|             7 | 2020-01-01 |   1000 | Credit |
|             8 | 2020-01-01 |   2000 | Credit |
|             9 | 2020-01-01 |    500 | Debit  |
|            10 | 2020-01-01 |   1000 | Credit |
|            11 | 2020-01-01 |    200 | Credit |
|            12 | 2020-01-01 |    100 | Debit  |
|            13 | 2020-01-01 |   1000 | Credit |
|            14 | 2020-01-01 |   2000 | Credit |
+---------------+------------+--------+--------+
14 rows in set (0.0051 sec)
```

```
SELECT * FROM users;
```

userId	firstName	lastName	email	subscriptionStatus	subscriptionFee	phone	registrationDate
1	John	Smith	john.smith@gmail.com	Premium	9.99	123-456-7890	2019-01-01
2	Jane	Doe	jane.doe@gmail.com	Premium	9.99	123-456-7890	2019-05-01
3	Bob	Smith	bob.smith@gmail.com	Premium	9.99	123-456-7890	2022-01-01
4	Alice	Doe	alice.doe@gmail.com	Premium	9.99	123-456-7890	2022-05-01
5	John	Doe	john.doe@gmail.com	Free	0	123-456-7890	2022-05-01
6	Jane	Smith	jane.smith@gmail.com	Free	0	123-456-7890	2022-05-01
7	Bob	Doe	bob.doe@gmail.com	Free	0	123-456-7890	2022-05-01

7 rows in set (0.0039 sec)

```
SELECT * FROM receives;
```

transactionId	recordId
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8

8 rows in set (0.0105 sec)

```
SELECT * FROM recordLabels;
```

recordId	name
1	Sony Music
2	Universal Music Group
3	Warner Music Group
4	EMI Music
5	BMG
6	Atlantic Records
7	Capitol Records
8	Republic Records
9	Def Jam Recordings
10	RCA Records

10 rows in set (0.0108 sec)

```
SELECT * FROM collaboratedWith;
```

```
+-----------+-----------+
| artistId1 | artistId2 |
+-----------+-----------+
|         1 |         2 |
|         1 |         3 |
|         1 |         4 |
|         2 |         3 |
|         2 |         5 |
|         3 |         1 |
|         3 |         4 |
|         4 |         1 |
|         4 |         5 |
|         5 |         2 |
|         6 |         3 |
|         6 |         4 |
+-----------+-----------+
12 rows in set (0.0107 sec)
```

```
SELECT * FROM artistType;
```

```
+--------------+----------+
| artistTypeId | type     |
+--------------+----------+
|            1 | Singer   |
|            2 | Musician |
|            3 | Producer |
|            4 | Lyricist |
|            5 | Composer |
+--------------+----------+
5 rows in set (0.0108 sec)
```

```
SELECT * FROM belongsTo;
```

```
+--------------+----------+
| artistTypeId | artistId |
+--------------+----------+
|            1 |        1 |
|            1 |        6 |
|            2 |        2 |
|            2 |        7 |
|            3 |        3 |
|            3 |        8 |
|            4 |        4 |
|            4 |        9 |
|            5 |        5 |
|            5 |       10 |
+--------------+----------+
10 rows in set (0.0108 sec)
```

```
SELECT * FROM sponsoredBy;
```

```
+-----------+-----------+
| sponsorId | podcastId |
+-----------+-----------+
|         1 |         1 |
|         2 |         2 |
|         3 |         3 |
|         4 |         4 |
+-----------+-----------+
4 rows in set (0.0108 sec)
```

```
SELECT * FROM sponsors;
```

```
+-----------+-----------+---------+
| sponsorId | name      | amount  |
+-----------+-----------+---------+
|         1 | Sponsor 1 | 1000.00 |
|         2 | Sponsor 2 | 2000.00 |
|         3 | Sponsor 3 | 2000.00 |
|         4 | Sponsor 4 | 4000.00 |
|         5 | Sponsor 5 | 4000.00 |
+-----------+-----------+---------+
5 rows in set (0.0119 sec)
```

```
SELECT * FROM createdBy;
```

```
+-----------+--------+
| podcastId | hostId |
+-----------+--------+
|         1 |      1 |
|         2 |      2 |
|         3 |      3 |
|         4 |      4 |
|         5 |      5 |
|         6 |      6 |
+-----------+--------+
6 rows in set (0.0107 sec)
```

```
SELECT * FROM specialGuests;
```

```
+---------+--------------+
| guestId | name         |
+---------+--------------+
|       1 | Jhon Doe     |
|       2 | James Smith  |
|       3 | John Smith   |
|       4 | Jhon Doe     |
|       5 | James Thanos |
|       6 | Bob Doe      |
+---------+--------------+
6 rows in set (0.0108 sec)
```

```
SELECT * FROM podcastHistory;
```

```
+-----------+-------+------+-------------+--------+
| podcastId | month | year | subscribers | rating |
+-----------+-------+------+-------------+--------+
|         1 |     1 | 2020 |        1000 |    4.5 |
|         1 |     2 | 2020 |        2000 |    4.5 |
|         1 |     3 | 2020 |        3000 |    4.5 |
|         1 |     4 | 2020 |        4000 |    4.5 |
|         2 |     1 | 2020 |        5000 |      0 |
|         2 |     2 | 2020 |        6000 |      3 |
|         2 |     3 | 2020 |        7000 |      3 |
+-----------+-------+------+-------------+--------+
7 rows in set (0.0038 sec)
```

```
SELECT * FROM podcastHas;
```

```
+-----------+---------+
| podcastId | genreId |
+-----------+---------+
|         1 |       1 |
|         1 |       2 |
|         1 |       4 |
|         2 |       3 |
|         2 |       7 |
|         2 |       8 |
|         3 |       3 |
|         3 |      11 |
+-----------+---------+
8 rows in set (0.0106 sec)
```

```
SELECT * FROM episodeFeaturesGuest;
```

```
+-----------+--------+---------+
| podcastId | number | guestId |
+-----------+--------+---------+
|         1 |      1 |       1 |
|         1 |      1 |       2 |
|         1 |      2 |       3 |
|         1 |      3 |       4 |
|         2 |      1 |       5 |
|         2 |      2 |       6 |
|         2 |      3 |       1 |
|         3 |      1 |       2 |
|         3 |      2 |       3 |
|         4 |      1 |       4 |
|         5 |      1 |       5 |
+-----------+--------+---------+
11 rows in set (0.0108 sec)
```

```
SELECT * FROM episodes;
```

podcastId	number	title	duration	releaseDate	ListeningCount	AdvertisementCount
1	1	Episode 1	1.5	2020-01-01	1000	10
1	2	Episode 2	1.5	2020-01-02	1000	10
1	3	Episode 3	1.5	2020-01-03	1000	10
2	1	Episode A	1.5	2020-01-04	1000	10
2	2	Episode B	1.5	2020-01-05	1000	10
2	3	Episode C	1.5	2020-01-06	1000	10
3	1	Episode XX	1.5	2020-01-07	1000	10
3	2	Episode YY	1.5	2020-01-08	1000	10
4	1	Episode 1	1.5	2020-01-09	1000	10
5	1	The Episode	1.5	2020-01-10	1000	10

10 rows in set (0.0108 sec)

```
SELECT * FROM podcasts;
```

podcastId	name	country	language	rating	episodeCount	totalSubscribers
1	The Joe Rogan Experience	USA	English	4.5	1000	1000000
2	The Daily	USA	English	4.5	1000	1000000
3	The Tim Ferriss Show	USA	English	4.5	1000	1000000
4	The Joe Budden Podcast	USA	English	4.5	1000	1000000
5	The Ben Shapiro Show	USA	English	4.5	1000	1000000
6	The Joe Rogan Experience	USA	English	4.5	1000	1000000
7	The Daily	USA	English	4.5	1000	1000000
8	The Tim Ferriss Show	USA	English	4.5	1000	1000000
9	The Joe Budden Podcast	USA	English	4.5	1000	1000000
10	The Ben Shapiro Show	USA	English	4.5	1000	1000000

10 rows in set (0.0106 sec)

```
SELECT * FROM creates;
```

artistId	songId	type
1	1	Guest
2	2	Collaborated
3	1	Guest
4	2	Guest
5	2	Collaborated
6	3	Guest
7	4	Guest
8	5	Collaborated
9	6	Collaborated
10	4	Collaborated

10 rows in set (0.0102 sec)

```
SELECT * FROM artistHas;
```

```
+-----------+----------+
| artistId | albumId |
+----------+---------+
|         1 |        1 |
|         1 |        2 |
|         2 |        3 |
|         3 |        1 |
|         3 |        4 |
|         4 |        5 |
|         6 |        6 |
|         8 |        3 |
|         9 |        1 |
|         9 |        2 |
+-----------+----------+
10 rows in set (0.0107 sec)
```

```
SELECT * FROM artists;
```

```
+----------+---------------+--------+---------------+-------------+------------------+----------+
| artistId | name          | status | country       | primaryGenre | monthlyListeners | recordId |
+----------+---------------+--------+---------------+-------------+------------------+----------+
|        1 | Lady Gaga     | Active | United States | Pop          |         10000000 |        1 |
|        2 | Kesha         | Active | United States | Pop          |         10000000 |        2 |
|        3 | Britney Spears | Active | United States | Pop          |         10000000 |        3 |
|        4 | Madonna       | Active | United States | Pop          |         10000000 |        4 |
|        5 | Rihanna       | Active | United States | Pop          |         10000000 |        5 |
|        6 | Ariana Grande | Active | United States | Pop          |         10000000 |        6 |
|        7 | Taylor Swift  | Active | United States | Pop          |         10000000 |        7 |
|        8 | Beyonce       | Active | United States | Pop          |         10000000 |        8 |
|        9 | Dua Lipa      | Active | United States | Pop          |         10000000 |        9 |
|       10 | Selena Gomez  | Active | United States | Pop          |         10000000 |       10 |
+----------+---------------+--------+---------------+-------------+------------------+----------+
10 rows in set (0.0108 sec)
```

```
SELECT * FROM songHas;
```

```
+--------+---------+
| songId | genreId |
+--------+---------+
|      1 |       1 |
|      2 |       1 |
|      3 |       4 |
|      4 |       5 |
|      5 |       6 |
|      6 |       1 |
|      7 |       5 |
|      8 |       9 |
|      9 |      11 |
|     10 |       2 |
+--------+---------+
10 rows in set (0.0103 sec)
```

```
SELECT * FROM genres;
```

```
+---------+-------------+
| genreId | name        |
+---------+-------------+
|       1 | Pop         |
|       2 | Rock        |
|       3 | Rap         |
|       4 | Country     |
|       5 | Electronic  |
|       6 | R&B         |
|       7 | Indie       |
|       8 | Alternative |
|       9 | Hip-Hop     |
|      10 | Folk        |
|      11 | Jazz        |
|      12 | Blues       |
+---------+-------------+
12 rows in set (0.0108 sec)
```

```
SELECT * FROM trackNumbers;
```

albumId	number
1	1
1	2
1	3
1	4
2	1
2	3
2	4
3	1
3	2
3	3
4	4
5	1
5	2
5	3
6	1
6	2
6	3

17 rows in set (0.0107 sec)

```
SELECT * FROM songs;
```

songId	royaltyRate	title	royaltyStatus	playCount	country	language	duration	primaryArtist	albumId
1	0.5	Just Dance	Active	10000000	United States	English	3.5	1	1
2	0.5	Poker Face	Active	10000000	United States	English	3.5	2	1
3	0.5	Bad Romance	Active	10000000	United States	English	3.5	3	1
4	0.5	Telephone	Active	10000000	United States	English	3.5	4	1
5	0.5	Alejandro	Active	10000000	United States	English	3.5	4	1
6	0.5	Born This Way	Active	10000000	United States	English	3.5	5	2
7	0.5	Applause	Active	10000000	United States	English	3.5	6	5
8	0.5	The Edge of Glory	Active	10000000	United States	English	3.5	6	8
9	0.5	Paparazzi	Active	10000000	United States	English	3.5	1	2
10	0.5	Bad Blood	Active	10000000	United States	English	3.5	7	3

10 rows in set (0.0108 sec)

```
SELECT * FROM albums;
```

```
+---------+-----------------+-------------+----------------+
| albumId | name            | releaseYear | edition        |
+---------+-----------------+-------------+----------------+
|       1 | The Fame        |        2008 | Deluxe Edition |
|       2 | The Fame Monster |        2009 | Deluxe Edition |
|       3 | Born This Way   |        2011 | Deluxe Edition |
|       4 | Artpop          |        2013 | Deluxe Edition |
|       5 | Joanne          |        2016 | Deluxe Edition |
|       6 | Chromatica      |        2020 | Deluxe Edition |
|       7 | The Fame Monster |        2009 | Deluxe Edition |
|       8 | Born This Way   |        2011 | Deluxe Edition |
|       9 | Artpop          |        2013 | Deluxe Edition |
|      10 | Joanne          |        2016 | Deluxe Edition |
+---------+-----------------+-------------+----------------+
10 rows in set (0.0100 sec)
```

4 SQL Queries

4.1 Task and Operations

Information Processing

- addSongDetail

```
SQL > INSERT INTO songs (royaltyRate, title, royaltyStatus,
playCount, country, language, duration, primaryArtist, albumId)
VALUES (0.5, "Something Just Like This", "Active", 10000000,
"United States", "English", 3.5, 6, 6);
Query OK, 1 row affected (0.0163 sec)

SQL > INSERT INTO artistHas(artistId, albumId) VALUES(4, 6);
Query OK, 1 row affected (0.0026 sec)

SQL > INSERT INTO creates (artistId, songId, type)
VALUES (4, 11, 'Guest');
Query OK, 1 row affected (0.0025 sec)

SQL > INSERT INTO songHas(songId, genreId) VALUES(11, 1);
Query OK, 1 row affected (0.0023 sec)
```

- addArtistDetail

```
SQL > INSERT INTO artists (name, status, country, primaryGenre,
monthlyListeners, recordId)
VALUES ("Drake", "Active", "United States", "Pop",
10000000, 1);
Query OK, 1 row affected (0.0030 sec)

 SQL > INSERT INTO creates(artistId, songId, type)
 VALUES(11, 2, "Guest");
Query OK, 1 row affected (0.0027 sec)

 SQL > INSERT INTO collaboratedWith (artistId1, artistId2)
 VALUES (2, 11);
Query OK, 1 row affected (0.0023 sec)

 SQL > INSERT INTO artistHas(artistId, albumId) VALUES(11, 6);
Query OK, 1 row affected (0.0029 sec)

SQL > INSERT INTO belongsTo(artistTypeId, artistId)
VALUES(1, 11);
Query OK, 1 row affected (0.0035 sec)
```

- addPodcastHostDetail

```
SQL > INSERT INTO podcastHosts (firstName, lastName, contact, email,
flatFee, city)
-> VALUES ("Joe", "Rogan", "8765645635", "joe.rogan@gmail.com",
2000.00, "Austin");
Query OK, 1 row affected (0.0039 sec)

 SQL > INSERT INTO createdBy (podcastId, hostId) VALUES (1, 7);
Query OK, 1 row affected (0.0037 sec)
```

- addPodcastEpDetail

```
SQL > INSERT INTO episodes (podcastId, number, title, duration,
releaseDate, ListeningCount, AdvertisementCount)
VALUES (1, 4, "Alex Jones Returns!", 1.5,
"2020-01-01", 1000, 10);
Query OK, 1 row affected (0.0037 sec)

 SQL > INSERT INTO episodeFeaturesGuest(podcastId, number, guestId)
 VALUES(1, 4, 5);
Query OK, 1 row affected (0.0038 sec)
```

- updateSongDetail

```
SQL > UPDATE songs SET royaltyRate=50, title="Shape Of You",
royaltyStatus="PAID", playCount=30, country="USA", language="ENGLISH",
duration=3.5, primaryArtist=1, albumId=2 WHERE songId=1;
Query OK, 1 row affected (0.0039 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- updateArtistDetail

```
SQL > UPDATE artists SET name="Ed Sheeran", status="Retired", country="USA",
primaryGenre="POP", monthlyListeners=35, recordId=2 WHERE artistId=1;
Query OK, 1 row affected (0.0040 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- updatePodcastHostDetail

```
SQL > UPDATE podcastHosts SET firstName="Jhon", lastName="Boa",
contact="7865678765", email="jboa@mail.com", flatFee=50, city="Raleigh"
WHERE HostId=1;
Query OK, 1 row affected (0.0038 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- updatePodcastEpDetail

```
SQL > UPDATE episodes SET title="The Power of Mindfulness", duration=6.0,
releaseDate='2020-01-01', ListeningCount=79, AdvertisementCount=14
WHERE podcastId = 1 AND number = 2;
Query OK, 1 row affected (0.0036 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- deleteSongDetail

```
SQL > DELETE FROM songs WHERE songId=11;
Query OK, 1 row affected (0.0040 sec)
```

- deleteArtistDetail

```
SQL > DELETE FROM artists WHERE artistId=11;
Query OK, 1 row affected (0.0042 sec)
```

- deletePodcastHostDetail

```
SQL > DELETE FROM podcastHosts WHERE hostId=7;
Query OK, 1 row affected (0.0046 sec)
```

- deletePodcastEpDetail

```
SQL > DELETE FROM episodes WHERE podcastId=1 AND number = 4;
Query OK, 1 row affected (0.0041 sec)
```

- assignSongToAlbum

```
UPDATE songs SET albumId=5 WHERE songId=1;
Query OK, 1 row affected (0.0043 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- assignArtistToAlbum

```
SQL > INSERT INTO artistHas (artistId, albumId) VALUES (10, 6);
Query OK, 1 row affected (0.0039 sec)
```

- assignArtistToRecordLabel

```
SQL > UPDATE artists SET recordId=3 WHERE artistId=1;
Query OK, 1 row affected (0.0388 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

- assignPodcastEpToPodcast

```
SQL > UPDATE episodes SET podcastId=10 WHERE podcastId = 1 AND number = 1;
Query OK, 1 row affected (0.0039 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

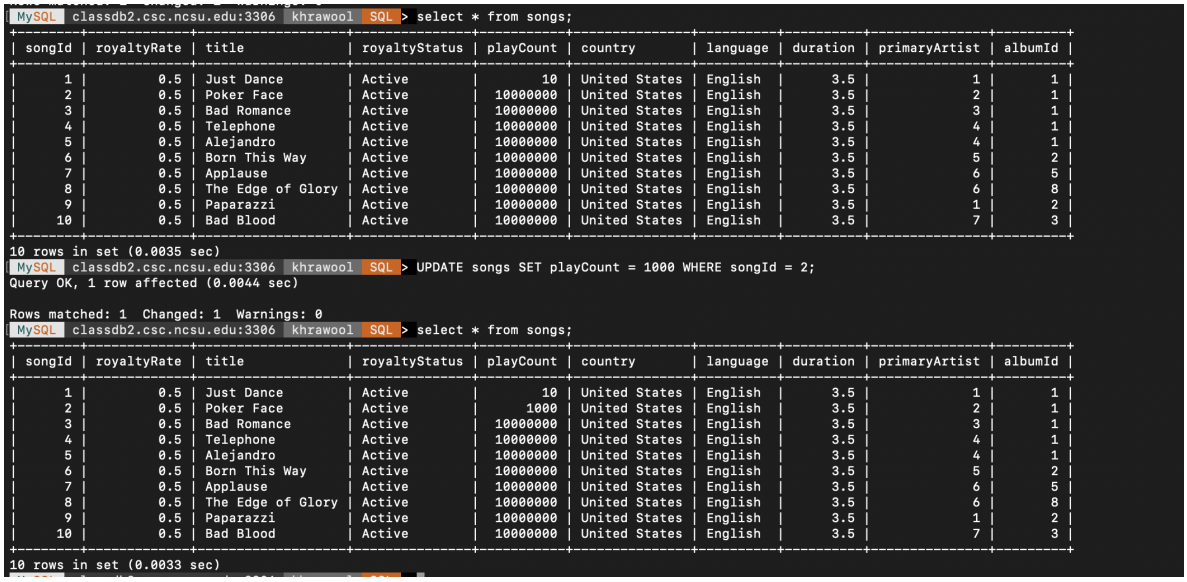
- assignPodcastHostToPodcast

```
SQL > INSERT INTO createdBy (podcastId, hostId) VALUES (4, 3);
Query OK, 1 row affected (0.0037 sec)
```

Maintaining metadata and records:

- addPlayCountSong

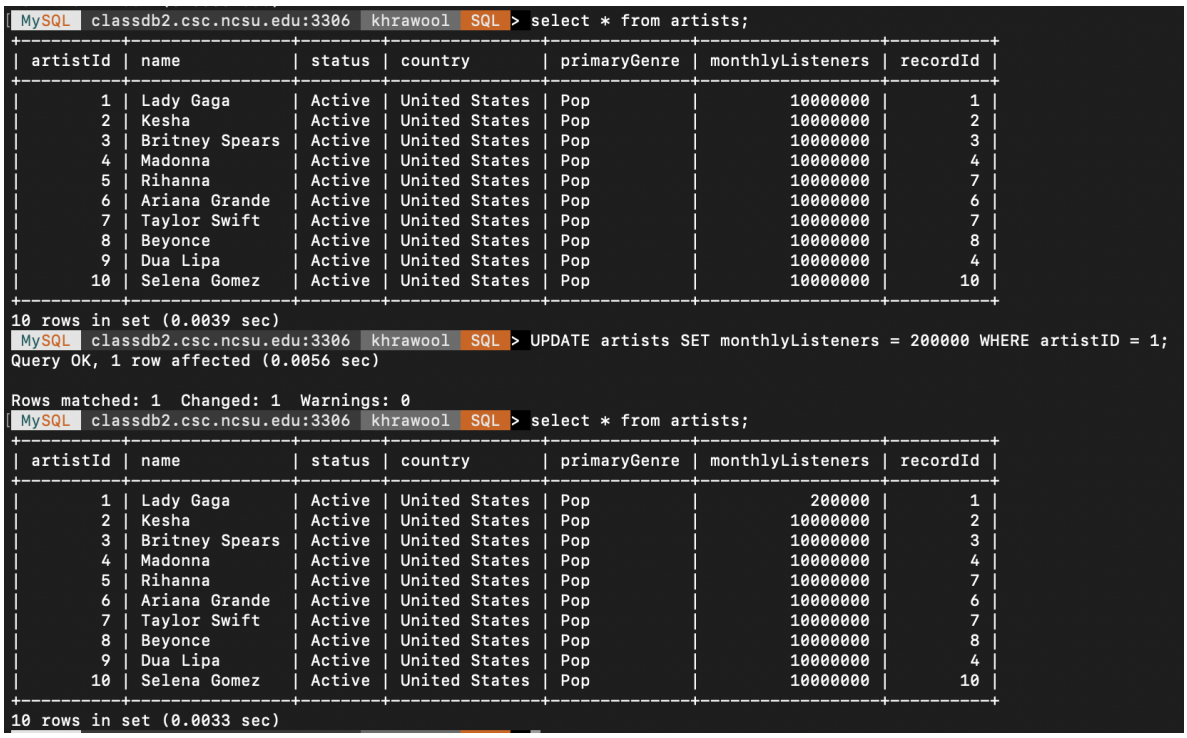
```
UPDATE songs SET playCount = 1000 WHERE songId = 2;
```



```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from songs;
+--------+-------------+-------------------+---------------+-----------+---------------+----------+----------+---------------+---------+
| songId | royaltyRate | title             | royaltyStatus | playCount | country       | language | duration | primaryArtist | albumId |
+--------+-------------+-------------------+---------------+-----------+---------------+----------+----------+---------------+---------+
|      1 |         0.5 | Just Dance        | Active        |        10 | United States | English  |      3.5 |             1 |       1 |
|      2 |         0.5 | Poker Face        | Active        |  10000000 | United States | English  |      3.5 |             2 |       1 |
|      3 |         0.5 | Bad Romance       | Active        |  10000000 | United States | English  |      3.5 |             3 |       1 |
|      4 |         0.5 | Telephone         | Active        |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      5 |         0.5 | Alejandro         | Active        |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      6 |         0.5 | Born This Way     | Active        |  10000000 | United States | English  |      3.5 |             5 |       2 |
|      7 |         0.5 | Applause          | Active        |  10000000 | United States | English  |      3.5 |             6 |       5 |
|      8 |         0.5 | The Edge of Glory | Active        |  10000000 | United States | English  |      3.5 |             6 |       8 |
|      9 |         0.5 | Paparazzi         | Active        |  10000000 | United States | English  |      3.5 |             1 |       2 |
|     10 |         0.5 | Bad Blood         | Active        |  10000000 | United States | English  |      3.5 |             7 |       3 |
+--------+-------------+-------------------+---------------+-----------+---------------+----------+----------+---------------+---------+
10 rows in set (0.0035 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE songs SET playCount = 1000 WHERE songId = 2;
Query OK, 1 row affected (0.0044 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from songs;
+--------+-------------+-------------------+---------------+-----------+---------------+----------+----------+---------------+---------+
| songId | royaltyRate | title             | royaltyStatus | playCount | country       | language | duration | primaryArtist | albumId |
+--------+-------------+-------------------+---------------+-----------+---------------+----------+----------+---------------+---------+
|      1 |         0.5 | Just Dance        | Active        |        10 | United States | English  |      3.5 |             1 |       1 |
|      2 |         0.5 | Poker Face        | Active        |      1000 | United States | English  |      3.5 |             2 |       1 |
|      3 |         0.5 | Bad Romance       | Active        |  10000000 | United States | English  |      3.5 |             3 |       1 |
|      4 |         0.5 | Telephone         | Active        |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      5 |         0.5 | Alejandro         | Active        |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      6 |         0.5 | Born This Way     | Active        |  10000000 | United States | English  |      3.5 |             5 |       2 |
|      7 |         0.5 | Applause          | Active        |  10000000 | United States | English  |      3.5 |             6 |       5 |
|      8 |         0.5 | The Edge of Glory | Active        |  10000000 | United States | English  |      3.5 |             6 |       8 |
|      9 |         0.5 | Paparazzi         | Active        |  10000000 | United States | English  |      3.5 |             1 |       2 |
|     10 |         0.5 | Bad Blood         | Active        |  10000000 | United States | English  |      3.5 |             7 |       3 |
+--------+-------------+-------------------+---------------+-----------+---------------+----------+----------+---------------+---------+
10 rows in set (0.0033 sec)
```

- addMonthlyListeners(artistId, monthlyListeners)

```
UPDATE artists SET monthlyListeners = 200000 WHERE artistID = 1;
```



```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from artists;
+----------+---------------+--------+---------------+--------------+------------------+----------+
| artistId | name          | status | country       | primaryGenre | monthlyListeners | recordId |
+----------+---------------+--------+---------------+--------------+------------------+----------+
|        1 | Lady Gaga     | Active | United States | Pop          |         10000000 |        1 |
|        2 | Kesha         | Active | United States | Pop          |         10000000 |        2 |
|        3 | Britney Spears| Active | United States | Pop          |         10000000 |        3 |
|        4 | Madonna       | Active | United States | Pop          |         10000000 |        4 |
|        5 | Rihanna       | Active | United States | Pop          |         10000000 |        7 |
|        6 | Ariana Grande | Active | United States | Pop          |         10000000 |        6 |
|        7 | Taylor Swift  | Active | United States | Pop          |         10000000 |        7 |
|        8 | Beyonce       | Active | United States | Pop          |         10000000 |        8 |
|        9 | Dua Lipa      | Active | United States | Pop          |         10000000 |        4 |
|       10 | Selena Gomez  | Active | United States | Pop          |         10000000 |       10 |
+----------+---------------+--------+---------------+--------------+------------------+----------+
10 rows in set (0.0039 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE artists SET monthlyListeners = 200000 WHERE artistID = 1;
Query OK, 1 row affected (0.0056 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from artists;
+----------+---------------+--------+---------------+--------------+------------------+----------+
| artistId | name          | status | country       | primaryGenre | monthlyListeners | recordId |
+----------+---------------+--------+---------------+--------------+------------------+----------+
|        1 | Lady Gaga     | Active | United States | Pop          |           200000 |        1 |
|        2 | Kesha         | Active | United States | Pop          |         10000000 |        2 |
|        3 | Britney Spears| Active | United States | Pop          |         10000000 |        3 |
|        4 | Madonna       | Active | United States | Pop          |         10000000 |        4 |
|        5 | Rihanna       | Active | United States | Pop          |         10000000 |        7 |
|        6 | Ariana Grande | Active | United States | Pop          |         10000000 |        6 |
|        7 | Taylor Swift  | Active | United States | Pop          |         10000000 |        7 |
|        8 | Beyonce       | Active | United States | Pop          |         10000000 |        8 |
|        9 | Dua Lipa      | Active | United States | Pop          |         10000000 |        4 |
|       10 | Selena Gomez  | Active | United States | Pop          |         10000000 |       10 |
+----------+---------------+--------+---------------+--------------+------------------+----------+
10 rows in set (0.0033 sec)
```

- addTotalSubscribers(podcastId, totalSubscribers)

```
UPDATE podcasts SET totalSubscribers = 500000 WHERE podcastId = 1;
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0039 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE podcasts SET totalSubscribers = 500000 WHERE podcastId = 1;
Query OK, 1 row affected (0.0050 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |           500000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0033 sec)
```

- addPodcastRatings(podcastId, rating)

```
UPDATE podcasts SET rating = 5 WHERE podcastId = 1;
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |           500000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0033 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE podcasts SET rating = 5 WHERE podcastId = 1;
Query OK, 1 row affected (0.0046 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |      5 |         1000 |           500000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0037 sec)
```

- addPodcastEpisodeListenerCount(podcastId, episodeId, listenerCount)

```
UPDATE episodes SET ListeningCount = 4000 WHERE podcastId = 1 AND number = 1;
```

```
10 rows in set (0.0037 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from episodes;
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
| podcastId | number | title       | duration | releaseDate | ListeningCount | AdvertisementCount |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
|         1 |      1 | Episode 1   |      1.5 | 2020-01-01  |           1000 |                 10 |
|         1 |      2 | Episode 2   |      1.5 | 2020-02-02  |           1000 |                 10 |
|         1 |      3 | Episode 3   |      1.5 | 2020-01-03  |           1000 |                 10 |
|         2 |      1 | Episode A   |      1.5 | 2020-02-04  |           1000 |                 10 |
|         2 |      2 | Episode B   |      1.5 | 2020-03-05  |           1000 |                 10 |
|         2 |      3 | Episode C   |      1.5 | 2020-01-06  |           1000 |                 10 |
|         3 |      1 | Episode XX  |      1.5 | 2020-10-07  |           1000 |                 10 |
|         3 |      2 | Episode YY  |      1.5 | 2020-01-08  |           1000 |                 10 |
|         4 |      1 | Episode 1   |      1.5 | 2020-12-09  |           1000 |                 10 |
|         5 |      1 | The Episode |      1.5 | 2020-01-10  |           1000 |                 10 |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
10 rows in set (0.0041 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE episodes SET ListeningCount = 4000 WHERE podcastId = 1 AND number = 1;
Query OK, 1 row affected (0.0050 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from episodes;
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
| podcastId | number | title       | duration | releaseDate | ListeningCount | AdvertisementCount |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
|         1 |      1 | Episode 1   |      1.5 | 2020-01-01  |           4000 |                 10 |
|         1 |      2 | Episode 2   |      1.5 | 2020-02-02  |           1000 |                 10 |
|         1 |      3 | Episode 3   |      1.5 | 2020-01-03  |           1000 |                 10 |
|         2 |      1 | Episode A   |      1.5 | 2020-02-04  |           1000 |                 10 |
|         2 |      2 | Episode B   |      1.5 | 2020-03-05  |           1000 |                 10 |
|         2 |      3 | Episode C   |      1.5 | 2020-01-06  |           1000 |                 10 |
|         3 |      1 | Episode XX  |      1.5 | 2020-10-07  |           1000 |                 10 |
|         3 |      2 | Episode YY  |      1.5 | 2020-01-08  |           1000 |                 10 |
|         4 |      1 | Episode 1   |      1.5 | 2020-12-09  |           1000 |                 10 |
|         5 |      1 | The Episode |      1.5 | 2020-01-10  |           1000 |                 10 |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
10 rows in set (0.0036 sec)
```

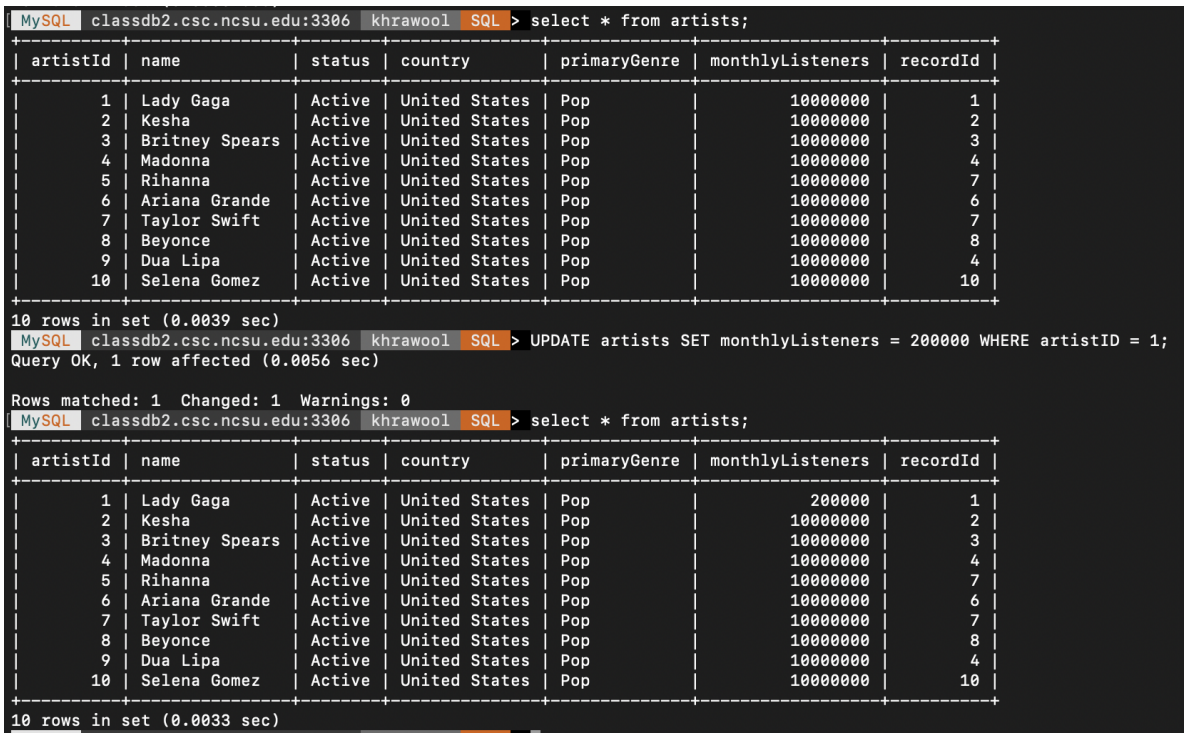
- updatePlayCountSong(songId, playCount)

```
UPDATE songs SET playCount = 1000 WHERE songId = 2;
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from songs;
+--------+-------------+-----------------+--------------+-----------+---------------+----------+----------+---------------+---------+
| songId | royaltyRate | title           | royaltyStatus | playCount | country       | language | duration | primaryArtist | albumId |
+--------+-------------+-----------------+--------------+-----------+---------------+----------+----------+---------------+---------+
|      1 |         0.5 | Just Dance      | Active       |        10 | United States | English  |      3.5 |             1 |       1 |
|      2 |         0.5 | Poker Face      | Active       |  10000000 | United States | English  |      3.5 |             2 |       1 |
|      3 |         0.5 | Bad Romance     | Active       |  10000000 | United States | English  |      3.5 |             3 |       1 |
|      4 |         0.5 | Telephone       | Active       |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      5 |         0.5 | Alejandro       | Active       |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      6 |         0.5 | Born This Way   | Active       |  10000000 | United States | English  |      3.5 |             5 |       2 |
|      7 |         0.5 | Applause        | Active       |  10000000 | United States | English  |      3.5 |             6 |       5 |
|      8 |         0.5 | The Edge of Glory | Active       |  10000000 | United States | English  |      3.5 |             6 |       8 |
|      9 |         0.5 | Paparazzi       | Active       |  10000000 | United States | English  |      3.5 |             1 |       2 |
|     10 |         0.5 | Bad Blood       | Active       |  10000000 | United States | English  |      3.5 |             7 |       3 |
+--------+-------------+-----------------+--------------+-----------+---------------+----------+----------+---------------+---------+
10 rows in set (0.0035 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE songs SET playCount = 1000 WHERE songId = 2;
Query OK, 1 row affected (0.0044 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from songs;
+--------+-------------+-----------------+--------------+-----------+---------------+----------+----------+---------------+---------+
| songId | royaltyRate | title           | royaltyStatus | playCount | country       | language | duration | primaryArtist | albumId |
+--------+-------------+-----------------+--------------+-----------+---------------+----------+----------+---------------+---------+
|      1 |         0.5 | Just Dance      | Active       |        10 | United States | English  |      3.5 |             1 |       1 |
|      2 |         0.5 | Poker Face      | Active       |      1000 | United States | English  |      3.5 |             2 |       1 |
|      3 |         0.5 | Bad Romance     | Active       |  10000000 | United States | English  |      3.5 |             3 |       1 |
|      4 |         0.5 | Telephone       | Active       |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      5 |         0.5 | Alejandro       | Active       |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      6 |         0.5 | Born This Way   | Active       |  10000000 | United States | English  |      3.5 |             5 |       2 |
|      7 |         0.5 | Applause        | Active       |  10000000 | United States | English  |      3.5 |             6 |       5 |
|      8 |         0.5 | The Edge of Glory | Active       |  10000000 | United States | English  |      3.5 |             6 |       8 |
|      9 |         0.5 | Paparazzi       | Active       |  10000000 | United States | English  |      3.5 |             1 |       2 |
|     10 |         0.5 | Bad Blood       | Active       |  10000000 | United States | English  |      3.5 |             7 |       3 |
+--------+-------------+-----------------+--------------+-----------+---------------+----------+----------+---------------+---------+
10 rows in set (0.0033 sec)
```

- updateMonthlyListeners(artistId, monthlyListeners)

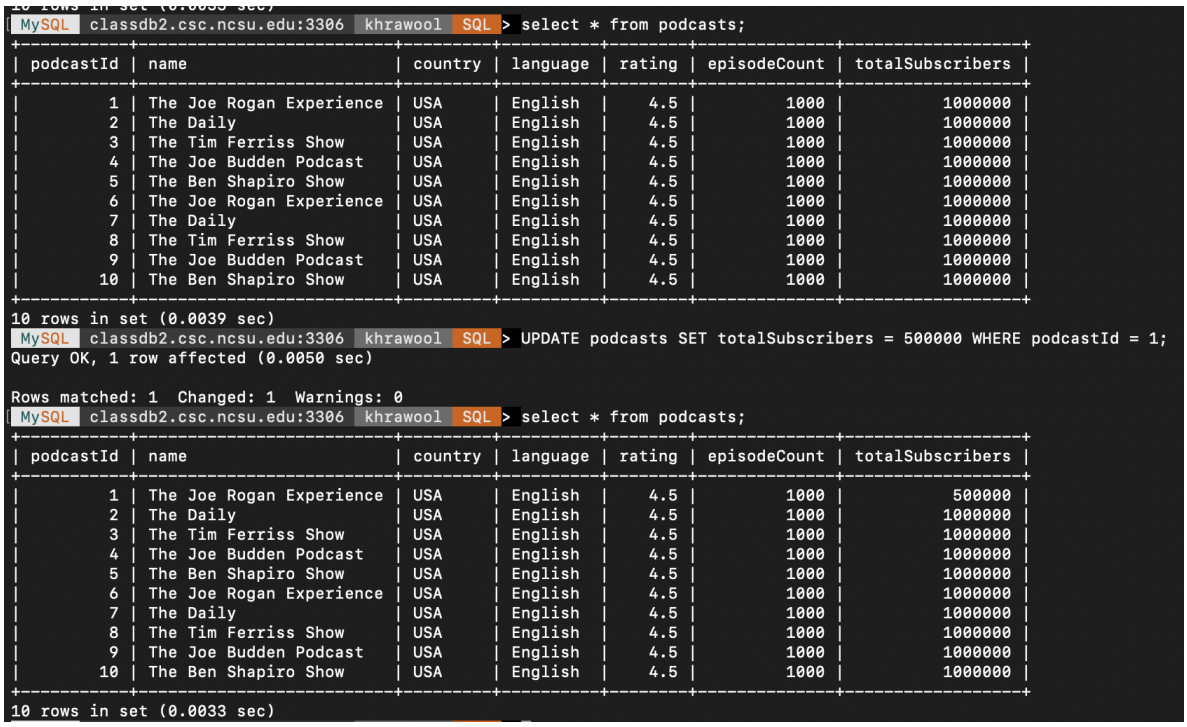
```
UPDATE artists SET monthlyListeners = 200000 WHERE artistID = 1;
```



```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from artists;
+----------+---------------+--------+---------------+--------------+------------------+----------+
| artistId | name          | status | country       | primaryGenre | monthlyListeners | recordId |
+----------+---------------+--------+---------------+--------------+------------------+----------+
|        1 | Lady Gaga     | Active | United States | Pop          |         10000000 |        1 |
|        2 | Kesha         | Active | United States | Pop          |         10000000 |        2 |
|        3 | Britney Spears| Active | United States | Pop          |         10000000 |        3 |
|        4 | Madonna       | Active | United States | Pop          |         10000000 |        4 |
|        5 | Rihanna       | Active | United States | Pop          |         10000000 |        7 |
|        6 | Ariana Grande | Active | United States | Pop          |         10000000 |        6 |
|        7 | Taylor Swift  | Active | United States | Pop          |         10000000 |        7 |
|        8 | Beyonce       | Active | United States | Pop          |         10000000 |        8 |
|        9 | Dua Lipa      | Active | United States | Pop          |         10000000 |        4 |
|       10 | Selena Gomez  | Active | United States | Pop          |         10000000 |       10 |
+----------+---------------+--------+---------------+--------------+------------------+----------+
10 rows in set (0.0039 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE artists SET monthlyListeners = 200000 WHERE artistID = 1;
Query OK, 1 row affected (0.0056 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from artists;
+----------+---------------+--------+---------------+--------------+------------------+----------+
| artistId | name          | status | country       | primaryGenre | monthlyListeners | recordId |
+----------+---------------+--------+---------------+--------------+------------------+----------+
|        1 | Lady Gaga     | Active | United States | Pop          |           200000 |        1 |
|        2 | Kesha         | Active | United States | Pop          |         10000000 |        2 |
|        3 | Britney Spears| Active | United States | Pop          |         10000000 |        3 |
|        4 | Madonna       | Active | United States | Pop          |         10000000 |        4 |
|        5 | Rihanna       | Active | United States | Pop          |         10000000 |        7 |
|        6 | Ariana Grande | Active | United States | Pop          |         10000000 |        6 |
|        7 | Taylor Swift  | Active | United States | Pop          |         10000000 |        7 |
|        8 | Beyonce       | Active | United States | Pop          |         10000000 |        8 |
|        9 | Dua Lipa      | Active | United States | Pop          |         10000000 |        4 |
|       10 | Selena Gomez  | Active | United States | Pop          |         10000000 |       10 |
+----------+---------------+--------+---------------+--------------+------------------+----------+
10 rows in set (0.0033 sec)
```

- updateTotalSubscribers(podcastId, totalSubscribers)

```
UPDATE podcasts SET totalSubscribers = 500000 WHERE podcastId = 1;
```



```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0039 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE podcasts SET totalSubscribers = 500000 WHERE podcastId = 1;
Query OK, 1 row affected (0.0050 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |           500000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0033 sec)
```

- UpdatePodcastRatings(podcastId, rating)

```
UPDATE podcasts SET rating = 5 WHERE podcastId = 1;
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |           500000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0033 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE podcasts SET rating = 5 WHERE podcastId = 1;
Query OK, 1 row affected (0.0046 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from podcasts;
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
| podcastId | name                     | country | language | rating | episodeCount | totalSubscribers |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
|         1 | The Joe Rogan Experience | USA     | English  |      5 |         1000 |           500000 |
|         2 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         3 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         4 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|         5 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         6 | The Joe Rogan Experience | USA     | English  |    4.5 |         1000 |          1000000 |
|         7 | The Daily                | USA     | English  |    4.5 |         1000 |          1000000 |
|         8 | The Tim Ferriss Show     | USA     | English  |    4.5 |         1000 |          1000000 |
|         9 | The Joe Budden Podcast   | USA     | English  |    4.5 |         1000 |          1000000 |
|        10 | The Ben Shapiro Show     | USA     | English  |    4.5 |         1000 |          1000000 |
+-----------+--------------------------+---------+----------+--------+--------------+------------------+
10 rows in set (0.0037 sec)
```

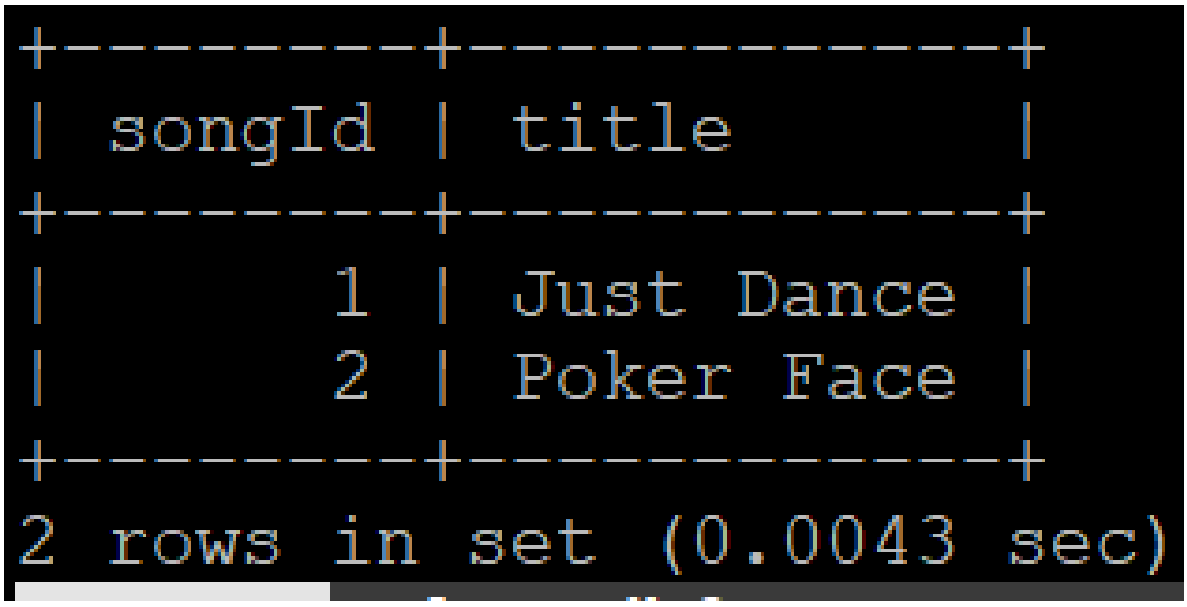
- updatePodcastEpisodeListenerCount(podcastId, episodeId, listenerCount)

```
UPDATE episodes SET ListeningCount = 4000 WHERE podcastId = 1 AND number = 1;
```

```
10 rows in set (0.0037 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from episodes;
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
| podcastId | number | title       | duration | releaseDate | ListeningCount | AdvertisementCount |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
|         1 |      1 | Episode 1   |      1.5 | 2020-01-01  |           1000 |                 10 |
|         1 |      2 | Episode 2   |      1.5 | 2020-02-02  |           1000 |                 10 |
|         1 |      3 | Episode 3   |      1.5 | 2020-01-03  |           1000 |                 10 |
|         2 |      1 | Episode A   |      1.5 | 2020-02-04  |           1000 |                 10 |
|         2 |      2 | Episode B   |      1.5 | 2020-03-05  |           1000 |                 10 |
|         2 |      3 | Episode C   |      1.5 | 2020-01-06  |           1000 |                 10 |
|         3 |      1 | Episode XX  |      1.5 | 2020-10-07  |           1000 |                 10 |
|         3 |      2 | Episode YY  |      1.5 | 2020-01-08  |           1000 |                 10 |
|         4 |      1 | Episode 1   |      1.5 | 2020-12-09  |           1000 |                 10 |
|         5 |      1 | The Episode |      1.5 | 2020-01-10  |           1000 |                 10 |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
10 rows in set (0.0041 sec)
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > UPDATE episodes SET ListeningCount = 4000 WHERE podcastId = 1 AND number = 1;
Query OK, 1 row affected (0.0050 sec)

Rows matched: 1  Changed: 1  Warnings: 0
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from episodes;
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
| podcastId | number | title       | duration | releaseDate | ListeningCount | AdvertisementCount |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
|         1 |      1 | Episode 1   |      1.5 | 2020-01-01  |           4000 |                 10 |
|         1 |      2 | Episode 2   |      1.5 | 2020-02-02  |           1000 |                 10 |
|         1 |      3 | Episode 3   |      1.5 | 2020-01-03  |           1000 |                 10 |
|         2 |      1 | Episode A   |      1.5 | 2020-02-04  |           1000 |                 10 |
|         2 |      2 | Episode B   |      1.5 | 2020-03-05  |           1000 |                 10 |
|         2 |      3 | Episode C   |      1.5 | 2020-01-06  |           1000 |                 10 |
|         3 |      1 | Episode XX  |      1.5 | 2020-10-07  |           1000 |                 10 |
|         3 |      2 | Episode YY  |      1.5 | 2020-01-08  |           1000 |                 10 |
|         4 |      1 | Episode 1   |      1.5 | 2020-12-09  |           1000 |                 10 |
|         5 |      1 | The Episode |      1.5 | 2020-01-10  |           1000 |                 10 |
+-----------+--------+-------------+----------+-------------+----------------+--------------------+
10 rows in set (0.0036 sec)
```

- findSongsByArtist(artistId)

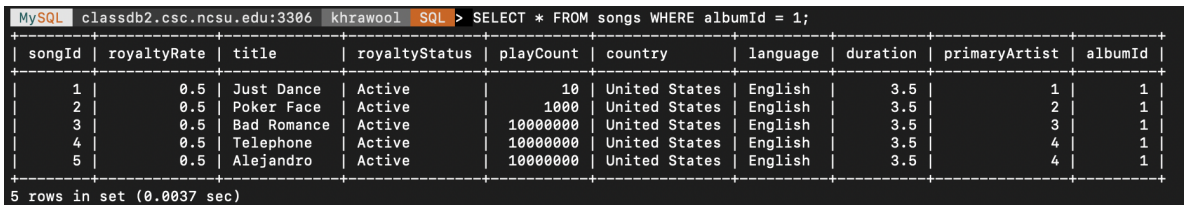
```
SELECT distinct(songs.songId), title FROM songs
LEFT JOIN creates ON songs.songId = creates.songId
WHERE artistId = 2 OR primaryArtist = 2;
```



```
+--------+--------------+
| songId | title        |
+--------+--------------+
|      1 | Just Dance   |
|      2 | Poker Face   |
+--------+--------------+
2 rows in set (0.0043 sec)
```

- findSongsByAlbum(albumId)

```
SELECT * FROM songs WHERE albumId = 1;
```



```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > SELECT * FROM songs WHERE albumId = 1;
+--------+-------------+------------+--------------+-----------+---------------+----------+----------+---------------+---------+
| songId | royaltyRate | title      | royaltyStatus | playCount | country       | language | duration | primaryArtist | albumId |
+--------+-------------+------------+--------------+-----------+---------------+----------+----------+---------------+---------+
|      1 |         0.5 | Just Dance | Active       |        10 | United States | English  |      3.5 |             1 |       1 |
|      2 |         0.5 | Poker Face | Active       |      1000 | United States | English  |      3.5 |             2 |       1 |
|      3 |         0.5 | Bad Romance | Active       |  10000000 | United States | English  |      3.5 |             3 |       1 |
|      4 |         0.5 | Telephone  | Active       |  10000000 | United States | English  |      3.5 |             4 |       1 |
|      5 |         0.5 | Alejandro  | Active       |  10000000 | United States | English  |      3.5 |             4 |       1 |
+--------+-------------+------------+--------------+-----------+---------------+----------+----------+---------------+---------+
5 rows in set (0.0037 sec)
```

- findPodcastEpisodes(podcastId)

```
SELECT * FROM episodes WHERE podcastId = 1;
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > SELECT * FROM episodes WHERE podcastId = 1;
+-----------+--------+-----------+----------+-------------+----------------+--------------------+
| podcastId | number | title     | duration | releaseDate | ListeningCount | AdvertisementCount |
+-----------+--------+-----------+----------+-------------+----------------+--------------------+
|         1 |      1 | Episode 1 |      1.5 | 2020-01-01  |           4000 |                 10 |
|         1 |      2 | Episode 2 |      1.5 | 2020-02-02  |           1000 |                 10 |
|         1 |      3 | Episode 3 |      1.5 | 2020-01-03  |           1000 |                 10 |
+-----------+--------+-----------+----------+-------------+----------------+--------------------+
3 rows in set (0.0061 sec)
```

Maintaining Payments

- makeRoyaltyPayment(songID)

```
SELECT s.songId, s.playCount, s.royaltyRate,
sh.playCount * s.royaltyRate AS amount
FROM songs s
INNER JOIN songHistory sh ON s.songId = sh.songId

SELECT recordId
FROM artists
WHERE artistId = (SELECT primaryArtist FROM songs WHERE songId = 2);

select count(*) from creates where songId = 2;

SELECT artistId
FROM creates
WHERE songId = 2
UNION
SELECT primaryArtist
FROM songs
WHERE songId = 2;

INSERT INTO serviceAccount (transactionId, date, amount, type)
VALUES (21, '2023-03-01', 1500.00, 'Debit');

INSERT INTO receives (transactionId, recordId) VALUES (21, 2);\
INSERT INTO distributesRoyalties (artistId, recordId, songId, date, amount)
VALUES (2, 2, 2, '2023-03-01', 262.5);
INSERT INTO distributesRoyalties (artistId, recordId, songId, date, amount)
VALUES (3, 2, 2, '2023-03-01', 262.5);
INSERT INTO distributesRoyalties (artistId, recordId, songId, date, amount)
VALUES (4, 2, 2, '2023-03-01', 262.5);
INSERT INTO distributesRoyalties (artistId, recordId, songId, date, amount)
VALUES (5, 2, 2, '2023-03-01', 262.5);
```

```
+--------+-----------+-------------+---------+
| songId | playCount | royaltyRate | amount  |
+--------+-----------+-------------+---------+
|      1 |        30 |          50 |    1500 |
|      2 |  10000000 |         0.5 | 5000000 |
|      3 |  10000000 |         0.5 | 5000000 |
|      4 |  10000000 |         0.5 | 5000000 |
|      5 |  10000000 |         0.5 | 5000000 |
|      6 |  10000000 |         0.5 | 5000000 |
|      7 |  10000000 |         0.5 | 5000000 |
|      8 |  10000000 |         0.5 | 5000000 |
|      9 |  10000000 |         0.5 | 5000000 |
|     10 |  10000000 |         0.5 | 5000000 |
|     14 |    500000 |         0.5 |  250000 |
|     15 |    500000 |         0.5 |  250000 |
+--------+-----------+-------------+---------+
12 rows in set (0.0026 sec)
```

```
+----------+
| recordId |
+----------+
|        2 |
+----------+
1 row in set (0.0026 sec)
```

```
+----------+
| count(*) |
+----------+
|        3 |
+----------+
1 row in set (0.0024 sec)
```

```
+----------+
| artistId |
+----------+
|        3 |
|        4 |
|        5 |
|        2 |
+----------+
4 rows in set (0.0027 sec)
```

transactionId	date	amount	type
1	2020-01-01	1000	Credit
2	2020-02-01	2000	Credit
3	2020-01-01	500	Debit
4	2020-01-01	1000	Credit
5	2020-03-01	200	Credit
6	2020-01-01	100	Debit
7	2020-02-01	1000	Credit
8	2020-01-01	2000	Credit
9	2020-01-01	500	Debit
10	2020-04-01	1000	Debit
11	2020-01-01	200	Debit
12	2020-03-01	100	Debit
13	2020-11-01	1000	Credit
14	2020-04-01	2000	Credit
15	2023-03-12	9.99	Premium
18	2023-03-12	9.99	Premium
19	2020-03-05	400	Debit
20	2023-03-12	100	Debit
657263	2023-03-12	9.99	Credit
888973	2023-03-12	9.99	Credit

- generateMonthlyRoyalties()

```
SELECT s.songId,
    (sh.playCount * s.royaltyRate) AS royaltyGenAmount
FROM songs s
JOIN songHistory sh ON s.songId = sh.songId
WHERE s.royaltyStatus = 'Not Active'
    AND sh.month = MONTH(CURRENT_DATE() - INTERVAL 1 MONTH)
    AND sh.year = YEAR(CURRENT_DATE() - INTERVAL 1 MONTH);
```

```
.royaltyStatus = 'Not Active'
+--------+------------------+
| songId | royaltyGenAmount |
+--------+------------------+
|     15 |              100 |
+--------+------------------+
1 row in set (0.0025 sec)
```

- makePaymentToPodcastHostForBonus()

```
SELECT createdBy.hostId, SUM(episodes.AdvertisementCount * 10) as bonus
FROM podcasts
JOIN createdBy ON podcasts.podcastId = createdBy.podcastId
JOIN episodes ON episodes.podcastId = podcasts.podcastId
WHERE MONTH(episodes.releaseDate) = MONTH(CURRENT_DATE - INTERVAL 1 MONTH)
AND YEAR(episodes.releaseDate) = YEAR(CURRENT_DATE - INTERVAL 1 MONTH)
GROUP BY createdBy.hostId;

INSERT INTO serviceAccount (transactionId, date, amount, type)
VALUES (19, '2020-03-05', 400.00, 'Debit');
INSERT INTO givesPaymentTo (transactionId, hostId)
VALUES (19, 6);
```

```
+--------+-------+
| hostId | bonus |
+--------+-------+
|      6 |   400 |
+--------+-------+
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from serviceAccount;
+---------------+------------+--------+--------+
| transactionId | date       | amount | type   |
+---------------+------------+--------+--------+
|             1 | 2020-01-01 |   1000 | Credit |
|             2 | 2020-02-01 |   2000 | Credit |
|             3 | 2020-01-01 |    500 | Debit  |
|             4 | 2020-01-01 |   1000 | Credit |
|             5 | 2020-03-01 |    200 | Credit |
|             6 | 2020-01-01 |    100 | Debit  |
|             7 | 2020-02-01 |   1000 | Credit |
|             8 | 2020-01-01 |   2000 | Credit |
|             9 | 2020-01-01 |    500 | Debit  |
|            10 | 2020-04-01 |   1000 | Debit  |
|            11 | 2020-01-01 |    200 | Debit  |
|            12 | 2020-03-01 |    100 | Debit  |
|            13 | 2020-11-01 |   1000 | Credit |
|            14 | 2020-04-01 |   2000 | Credit |
|            19 | 2020-03-05 |    400 | Debit  |
|            20 | 2023-03-12 |    100 | Debit  |
+---------------+------------+--------+--------+
16 rows in set (0.0032 sec)
```

```
MySQL classdb2.csc.ncsu.edu:3306 khrawool SQL > select * from givesPaymentTo;
+---------------+--------+
| transactionId | hostId |
+---------------+--------+
|             1 |      1 |
|             2 |      2 |
|             3 |      3 |
|             4 |      4 |
|             5 |      3 |
|             6 |      2 |
|             7 |      1 |
|            19 |      6 |
|            20 |      1 |
+---------------+--------+
9 rows in set (0.0030 sec)
```

- makePaymentToPodcastHostsForFlatFee()

```
SELECT
  podcastHosts.flatFee * COUNT(episodes.number) AS FlatFeePerPodcast
FROM
  podcastHosts
  JOIN createdBy ON podcastHosts.hostId = createdBy.hostId
  JOIN podcasts ON createdBy.podcastId = podcasts.podcastId
  JOIN episodes ON podcasts.podcastId = episodes.podcastId
WHERE
  episodes.releaseDate >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
GROUP BY
  podcasts.podcastId;
```

```
+--------------------+
| FlatFeePerPodcast |
+--------------------+
|             100.00 |
+--------------------+
1 row in set (0.0034 sec)
```

transactionId	date	amount	type
1	2020-01-01	1000	Credit
2	2020-02-01	2000	Credit
3	2020-01-01	500	Debit
4	2020-01-01	1000	Credit
5	2020-03-01	200	Credit
6	2020-01-01	100	Debit
7	2020-02-01	1000	Credit
8	2020-01-01	2000	Credit
9	2020-01-01	500	Debit
10	2020-04-01	1000	Debit
11	2020-01-01	200	Debit
12	2020-03-01	100	Debit
13	2020-11-01	1000	Credit
14	2020-04-01	2000	Credit
20	2023-03-12	100	Debit

15 rows in set (0.0032 sec)

transactionId	hostId
1	1
2	2
3	3
4	4
5	3
6	2
7	1
20	1

8 rows in set (0.0036 sec)

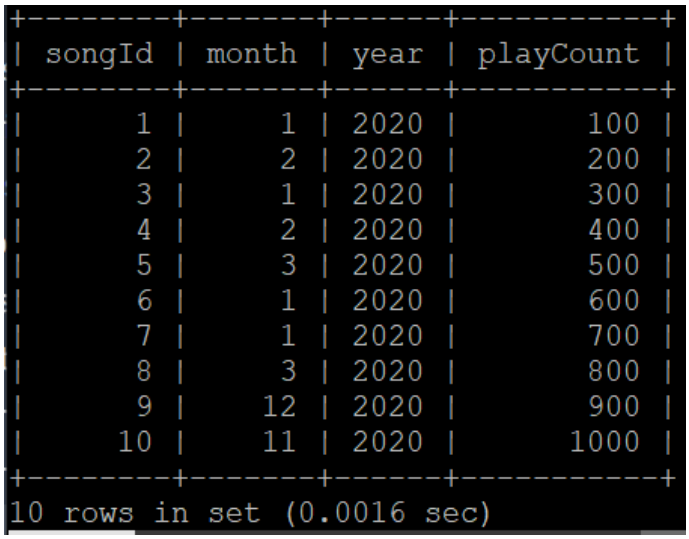
- receivePaymentFromSubscribers()

```
 SQL > INSERT INTO serviceAccount (date, amount, type)
SELECT CURDATE(), subscriptionFee, 'Credit'
FROM users
WHERE subscriptionStatus = 'Premium';
Query OK, 3 rows affected (0.0053 sec)
Records: 3  Duplicates: 0  Warnings: 0

INSERT INTO pays (userId, transactionId) VALUES (15, 657263);
Query OK, 1 row affected (0.0033 sec)

INSERT INTO pays (userId, transactionId) VALUES (18, 888973);
Query OK, 1 row affected (0.0036 sec)
```

Reports

- getPlayCountPerSongPerMonth

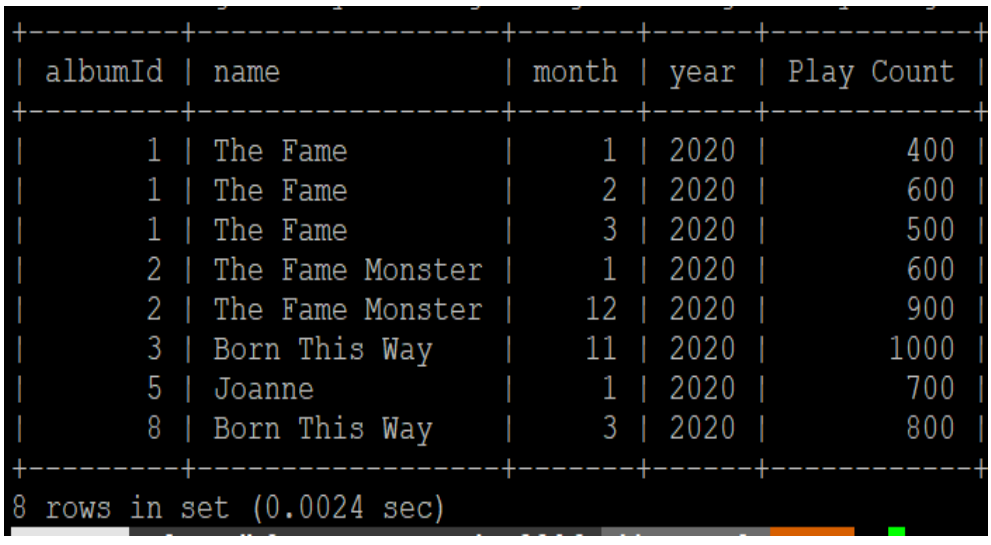
```
SELECT * FROM songHistory;
```



```
+--------+-------+------+-----------+
| songId | month | year | playCount |
+--------+-------+------+-----------+
|      1 |     1 | 2020 |       100 |
|      2 |     2 | 2020 |       200 |
|      3 |     1 | 2020 |       300 |
|      4 |     2 | 2020 |       400 |
|      5 |     3 | 2020 |       500 |
|      6 |     1 | 2020 |       600 |
|      7 |     1 | 2020 |       700 |
|      8 |     3 | 2020 |       800 |
|      9 |    12 | 2020 |       900 |
|     10 |    11 | 2020 |      1000 |
+--------+-------+------+-----------+
10 rows in set (0.0016 sec)
```

- getPlayCountPerAlbumPerMonth

```
SELECT albums.albumId, albums.name, month, year,
SUM(songHistory.playCount) AS "Play Count"
FROM songs
INNER JOIN songHistory
ON songs.songId = songHistory.songId
INNER JOIN albums ON songs.albumId = albums.albumId
GROUP BY albums.albumId, albums.name, month, year;
```



```
+---------+------------------+-------+------+------------+
| albumId | name             | month | year | Play Count |
+---------+------------------+-------+------+------------+
|       1 | The Fame         |     1 | 2020 |        400 |
|       1 | The Fame         |     2 | 2020 |        600 |
|       1 | The Fame         |     3 | 2020 |        500 |
|       2 | The Fame Monster |     1 | 2020 |        600 |
|       2 | The Fame Monster |    12 | 2020 |        900 |
|       3 | Born This Way    |    11 | 2020 |       1000 |
|       5 | Joanne           |     1 | 2020 |        700 |
|       8 | Born This Way    |     3 | 2020 |        800 |
+---------+------------------+-------+------+------------+
8 rows in set (0.0024 sec)
```

- getPlayCountPerArtistPerMonth

```
SELECT combined.artistId, combined.name, month, year, SUM(playCount)
FROM(
(SELECT artists.artistId, artists.name, month, year,
SUM(songHistory.playCount) AS playCount
FROM songs
INNER JOIN songHistory
ON songs.songId = songHistory.songId
INNER JOIN artists ON songs.primaryArtist = artists.artistId
GROUP BY artists.artistId, month, year)
UNION ALL
(SELECT artists.artistId, artists.name, month, year,
SUM(songHistory.playCount) AS playCount
FROM songHistory
INNER JOIN creates
ON creates.songId = songHistory.songId
INNER JOIN artists ON creates.artistId = artists.artistId
GROUP BY artists.artistId, month, year)) AS combined
GROUP BY combined.artistId, combined.name, month, year;
```

artistId	name	month	year	SUM(playCount)
1	Lady Gaga	1	2020	100
1	Lady Gaga	12	2020	900
2	Kesha	1	2020	100
2	Kesha	2	2020	200
3	Britney Spears	1	2020	400
3	Britney Spears	2	2020	200
4	Madonna	2	2020	600
4	Madonna	3	2020	500
5	Rihanna	1	2020	600
5	Rihanna	2	2020	200
6	Ariana Grande	1	2020	1000
6	Ariana Grande	3	2020	800
7	Taylor Swift	2	2020	400
7	Taylor Swift	2	2023	100
7	Taylor Swift	11	2020	1000
8	Beyonce	1	2023	400
8	Beyonce	2	2023	500
8	Beyonce	3	2020	500
8	Beyonce	3	2023	200
9	Dua Lipa	1	2020	600
10	Selena Gomez	2	2020	400

- calculateHostPayments

```
SELECT SUM(amount) FROM givesPaymentTo
INNER JOIN serviceAccount
ON givesPaymentTo.transactionId = serviceAccount.transactionId
WHERE date BETWEEN "2020-01-01" and "2020-02-02";
```

```
+-------------+
| SUM(amount) |
+-------------+
|        5600 |
+-------------+
1 row in set (0.0019 sec)
```

- calculateArtistPayments

```
SELECT SUM(amount) FROM distributesRoyalties
WHERE date BETWEEN "2020-01-01" and "2020-02-02";
```

```
+-------------+
| SUM(amount) |
+-------------+
|     1400.00 |
+-------------+
1 row in set (0.0016 sec)
```

- calculateRecordLabelPayments

```
SELECT 0.3 * SUM(amount) AS "Record Label Payments" FROM receives
INNER JOIN serviceAccount
ON receives.transactionId = serviceAccount.transactionId
WHERE date BETWEEN "2020-01-01" and "2020-02-02";
```

```
+----------------------+
| Record Label Payments |
+----------------------+
|                  390 |
+----------------------+
1 row in set (0.0023 sec)
```

- calculateMonthlyRevenue

```
SELECT YEAR(date) AS year, MONTH(date) AS month,
SUM(CASE WHEN type = "Credit" THEN amount ELSE 0 END) AS revenue
FROM serviceAccount
GROUP BY YEAR(date), MONTH(date);
```

```
mysql> SELECT YEAR(date) AS year, MONTH(date) AS month,
    -> SUM(CASE WHEN type = "Credit" THEN amount ELSE 0 END) AS revenue
    -> FROM serviceAccount
    -> GROUP BY YEAR(date), MONTH(date);
+------+-------+--------------------+
| year | month | revenue            |
+------+-------+--------------------+
| 2020 |     1 |               4000 |
| 2020 |     2 |               3000 |
| 2020 |     3 |                200 |
| 2020 |     4 |               2000 |
| 2020 |    11 |               1000 |
| 2023 |     3 | 129.86999702453613 |
+------+-------+--------------------+
6 rows in set (0.08 sec)
```

- calculateYearlyRevenue

```
SELECT YEAR(date) AS year,
SUM(CASE WHEN type = "Credit" THEN amount ELSE 0 END) AS revenue
FROM serviceAccount
GROUP BY YEAR(date);
```

```
mysql> SELECT YEAR(date) AS year,
    -> SUM(CASE WHEN type = "Credit" THEN amount ELSE 0 END) AS revenue
    -> FROM serviceAccount
    -> GROUP BY YEAR(date);
+------+--------------------+
| year | revenue            |
+------+--------------------+
| 2020 |              10200 |
| 2023 | 129.86999702453613 |
+------+--------------------+
2 rows in set (0.09 sec)
```

- getSongsByArtist

```
SELECT distinct(songs.songId), title FROM songs
LEFT JOIN creates ON songs.songId = creates.songId
WHERE artistId = 2 OR primaryArtist = 2;
```

```
+--------+------------+
| songId | title      |
+--------+------------+
|      1 | Just Dance |
|      2 | Poker Face |
+--------+------------+
2 rows in set (0.0043 sec)
```

- getSongsByAlbum

```
SELECT title from songs
WHERE albumId = 1;
```

```
+--------------+
| title        |
+--------------+
| Just Dance   |
| Poker Face   |
| Bad Romance  |
| Telephone    |
| Alejandro    |
+--------------+
5 rows in set (0.0019 sec)
```

- getPodcastEpisodesByPodcast

```
SELECT * FROM episodes
WHERE podcastId = 2;
```

```
+--------+-----------+----------+-------------+
| number | title     | duration | releaseDate |
+--------+-----------+----------+-------------+
|      1 | Episode A |      1.5 | 2020-02-04  |
|      2 | Episode B |      1.5 | 2020-03-05  |
|      3 | Episode C |      1.5 | 2020-01-06  |
+--------+-----------+----------+-------------+
3 rows in set (0.0018 sec)
```

4.2 EXPLAIN Query and INDEX

1. SQL Query 1

```
SELECT s.songId,
(sh.playCount * s.royaltyRate) AS royaltyGenAmount
FROM songs s
JOIN songHistory sh ON s.songId = sh.songId
WHERE s.royaltyStatus = 'Not Active'
AND sh.month = MONTH(CURRENT_DATE() - INTERVAL 1 MONTH)
AND sh.year = YEAR(CURRENT_DATE() - INTERVAL 1 MONTH);
```

2. Execution Plan

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	sh	ALL	PRIMARY	NULL	NULL	NULL	12	Using where
1	SIMPLE	s	eq_ref	PRIMARY	PRIMARY	4	khrawool.sh.songId	1	Using where

3. Index Creation Statement

```
create index royaltyIndex ON songs(royaltyStatus);
```

4. Execution Plan

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	s	ref	PRIMARY,royaltyIndex	royaltyIndex	257	const	2	Using index condition
1	SIMPLE	sh	eq_ref	PRIMARY	PRIMARY	12	khrawool.s.songId,const,const	1	Using where

1. SQL Query 2

```
Task and operation query:
INSERT INTO serviceAccount (date, amount, type)
SELECT CURDATE(), subscriptionFee, 'Credit'
FROM users
WHERE subscriptionStatus = 'Premium';

Using the Inner query:
SELECT CURDATE(), subscriptionFee, 'Credit'
FROM users
WHERE subscriptionStatus = 'Premium';
```

2. Execution Plan

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	users	ALL	NULL	NULL	NULL	NULL	8	Using where

3. Index Creation Statement

```
create index subscriptionIndex ON users(subscriptionStatus);
```

4. Execution Plan

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	users	ref	subscriptionIndex	subscriptionIndex	257	const	3	Using index condition

4.3 Correctness Proof

1.
```
SELECT SUM(amount) FROM givesPaymentTo
INNER JOIN serviceAccount
ON givesPaymentTo.transactionId = serviceAccount.transactionId
WHERE date BETWEEN "2020-01-01" and "2020-02-02";
```

Specification: returns the total amount paid to the Podcast Hosts between the dates 2020-01-01 and 2020-02-02

Relational Algebra:

$$\gamma_{\text{sum(amount)}}(\sigma_{\text{date BETWEEN "2020-01-01" and "2020-02-02"}}(\text{givesPaymentTo} \bowtie_{\text{givesPaymentTo.transactionId = serviceAccount.transactionId}} \text{serviceAccount}))$$

Correctness proof: "Suppose p is any tuple in the givesPaymentTo relation, and s is any tuple in the serviceAccount relation, such that the value p.transactionId is the same as the value s.transactionId. Each such combination of tuples (p, s)gives podcastHostId associated with the information of the transaction. For each such combination (p, s), the query checks if the date is between 2020-01-01 and 2020-02-02. The query then sums all the values present in the amount attribute. But this is exactly what our query should return; see the specification."

2.
```
SELECT albums.albumId, albums.name, month, year,
SUM(songHistory.playCount) AS "Play Count"
FROM songs
INNER JOIN songHistory
ON songs.songId = songHistory.songId
INNER JOIN albums ON songs.albumId = albums.albumId
GROUP BY albums.albumId, albums.name, month, year;
```

Specification: returns play count per album per month

Relational Algebra:

$$\rho_{\text{albumId, name, month, year, Play Count}}(\gamma_{\text{albums.albumId, albums.name, month, year, SUM(songHistory.playCount)}}((\text{songs} \bowtie_{\text{songs.songId = songHistory.songId}} \text{songHistory}) \bowtie_{\text{songs.albumId = albums.albumId}} \text{albums}))$$

Correctness proof: "Suppose s is any tuple in the songs relation, and h is any tuple in the songHistory relation, such that the value s.songId is the same as the value h.songId. Each such combination of tuples (s, h)gives song information about one song together with all the song history information for that song. Suppose a is any tuple in the albums relation, such that a.songId is the same as the value s.songId from above tuple. Each such combination of tuples (s, h, a) gives the song information with history about one song together with the album information for that song. For each such combination (s, h, a), the query groups the result with respect to the value of albumId, album name, month and year and returns the sum of playCount for each group with all the values which it grouped on. These values are the play count for each album for each month. But this is exactly what our query should return; see the specification."