# CPSC 304 Project Cover Page

Milestone #2

Date: 07-25-2023

Group Number: 32

| Name | Student Number | CS Alias (Userid) | Preferred E-mail Address |
|---|---|---|---|
| Jiawei Liu | 55362669 | b7j6x | 1943743535@qq.com |
| Flora Deng | 14085211 | d1i2t | floraa817@gmail.com |
| Tammie Liang | 52445806 | c1g1c | tammieliang@hotmail.com |

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

# ER Diagram

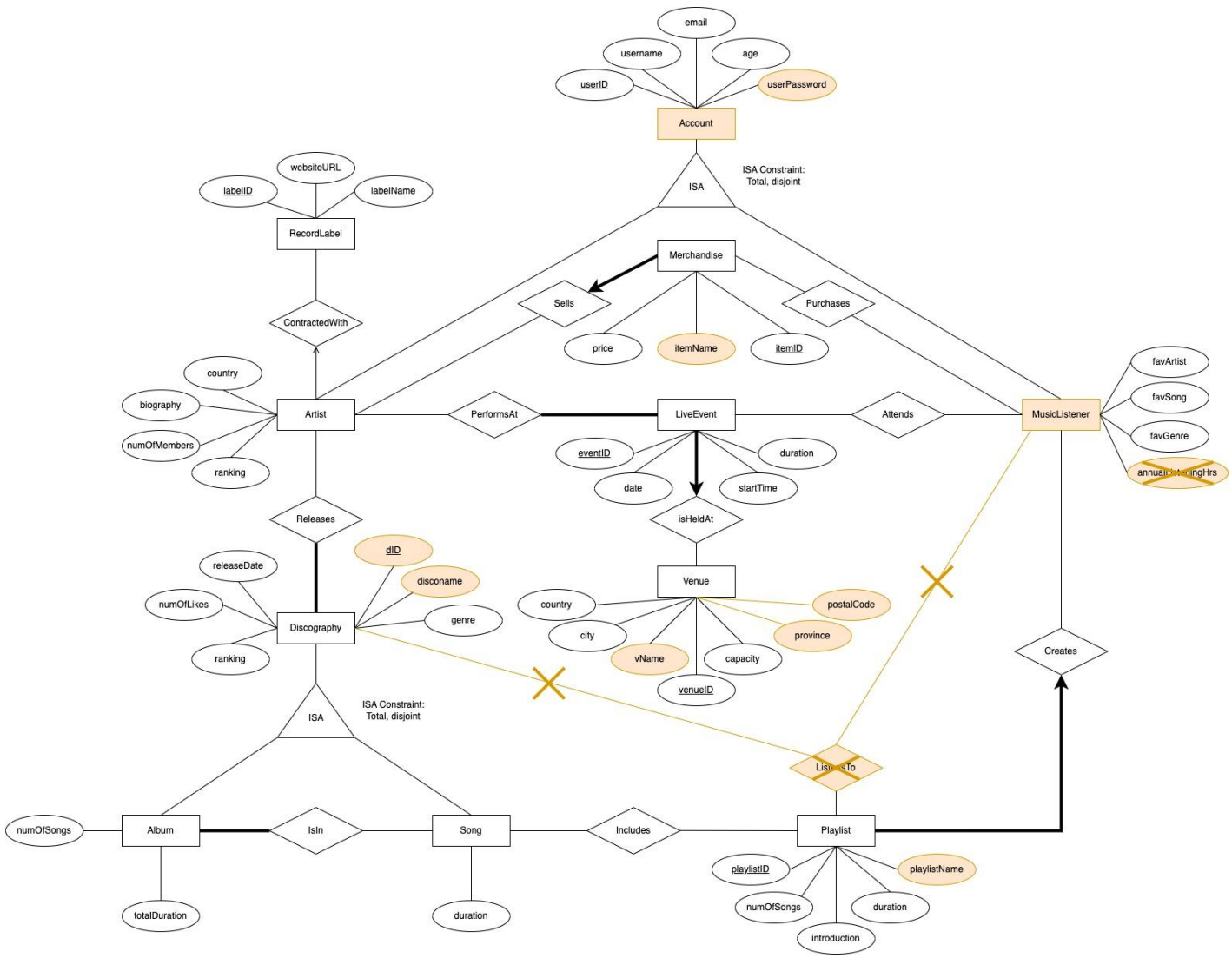


Diagram Link: (included in submission comments ☺)

ER diagram changes (marked in orange)

- Venue: added attributes postalCode and province
  - Reason: To add meaningful attributes that allow for normalization
- User: renamed the entity to “Account”, renamed the attribute “password” to “userPassword”
  - Reason: SQL Server will not allow an entity to be named “User”, and an attribute to be named “password”
- Listener: renamed the entity to “MusicListener”

    - Reason: SQL Server will not allow an entity to be named “Listener”
- Discography: rename ID to dID
    - Reason: Easier to distinguish from other entities' IDs
- Remove annualListeningHrs attribute
    - Reason: Not an important statistic, but requires maintenance and some sort of internal clock for each user
- Remove ListensTo relationship
    - Reason: For this project, it did not make sense to store what a listener would be listening to – it is not information we need to remember. We are not implementing a real-time recommendations algorithm or anything of that sort

We also renamed any “name” attribute to “[something]Name’ because SQL server does not allow us to name attributes “name”. We also renamed any “date” attribute to “[something]Date” since date is a data type

## Relational Schemas PRE-NORMALIZATION

- Artist_ContractedWith(artistID**:** string, username: string, email: string, age: integer, userPassword: string, country: string, biography: string, numOfMembers: integer, ranking: integer, **labelID: string**)
  PK: artistID
  CK: email
  FK: labelID

- MusicListener(listenerID**:** string, username: string, email: string, age: integer, userPassword: string, favArtist: string, favSong: string, favGenre: string)
  PK: listenerID
  CK: email
  FK: None

- RecordLabel(labelID: string, websiteURL: string, labelName: string)
  PK: labelID
  CK: websiteURL
  FK: None

- Discography(dID: string, discoName: string, genre: string, releaseDate: date, numOfLikes: integer, ranking: integer)
  PK: dID
  CK: None
  FK: None

- Album(**albumID:** string, numOfSongs: integer, totalDuration: time)
  PK: albumID
  CK: None
  FK: albumID

- Song(**songID:** string, duration: time)
  PK: songID
  CK: None
  FK: songID

- Playlist_Created(playlistID: string, **listenerID:** string, playlistName: string, numOfSongs: integer, introduction: string, duration: time)
  PK: playlistID
  CK: None

FK: listenerID

- Merchandise_Sold(<u>itemID:</u> string, **artistID:** string, itemName: string, price: double) PK: itemID
  CK: None
  FK: artistID

- Venue(<u>venueID:</u> string, country: string, postalCode: string, city: string, province: string, vName: string, capacity: string)
  PK: venueID
  CK: None
  FK: None

- LiveEvent_IsHeldAt(<u>eventID:</u> string, **venueID:** string, date: string, startTime: string, duration: double)
  PK: eventID
  CK: None
  FK: venueID

- IsIn(<u>**albumID:**</u> string, <u>**songID:**</u> string)
  PK: {albumID, songID}
  CK: None
  FK: albumID, songID

- Includes(<u>**songID:**</u> string, <u>**playlistID:**</u> string)
  PK: {songID, playlistID}
  CK: None
  FK: songID, playlistID

- PerformsAt(<u>**artistID:**</u> string, <u>**eventID:**</u> string)
  PK: {artistID, eventID}
  CK: None
  FK: artistID, eventID

- Purchases(<u>**listenerID:**</u> string, <u>**itemID:**</u> string)
  PK: {listenerID, itemID}
  CK: None
  FK: listenerID, itemID

- Attends(<u>**listenerID:**</u> string, <u>**eventID:**</u> string)

PK: {eventID, listenerID}
CK: None
FK: eventID, listenerID

- Releases(**<u>artistID:</u>** string, **<u>dID:</u>** string)
  PK: {artistID, dID}
  CK: None
  FK: artistID, dID

## Functional Dependencies PRE-NORMALIZATION

Artist_ContractedWith(artistID, username, email, age, userPassword, country, biography, numOfMembers, ranking, **labelID**)

- artistID -> username, email, age, userPassword, country, biography, numOfMembers, ranking
- email -> artistID, username, age, userPassword, country, biography, numOfMembers, ranking

MusicListener(listenerID, username, email, age, userPassword, favArtist, favSong, favGenre)

- listenerID -> username, email, age, userPassword, favArtist, favSong, favGenre
- email -> listenerID, username, age, userPassword, favArtist, favSong, favGenre

RecordLabel(labelID, websiteURL, labelName)

- labelID -> websiteURL, labelName
- websiteURL -> labelID, labelName

Normalized below Discography(dID, discoName, genre, releaseDate, numOfLikes, ranking)

- dID -> discoName, genre, releaseDate, numOfLikes, ranking
- numOfLikes -> ranking

Album(**albumID**, numOfSongs, totalDuration)

- albumID -> numOfSongs, totalDuration

Song(**songID**, duration)

- songID -> duration

Playlist_Created(playlistID, **listenerID**, playlistName, numOfSongs, introduction, duration)

- playlistID -> listenerID, playlistName, numOfSongs, introduction, duration

Merchandise_Sold(itemID, **artistID**, itemName, price):

- itemID -> artistID, itemName, price

Normalized below Venue(venueID, country, postalCode, city, province, vName, capacity)

- venueID -> country, postalCode, city, province, vName, capacity
- country, postalCode -> city, province

LiveEvent_IsHeldAt(eventID, **venueID**, eventDate, startTime, duration):

- eventID -> venueID, eventDate, startTime, duration

IsIn(**albumID**, **songID**):

- albumID, songID -> albumID, songID

Includes(**songID**, **playlistID**):

- songID, playlistID -> songID, playlistID

PerformsAt(**artistID**, **eventID**):

- artistID, eventID -> artistID, eventID

Purchases(**listenerID**, **itemID**):

- listenerID, itemID -> listenerID, itemID

Attends(**listenerID, eventID**):

- eventID, listenerID -> eventID, listenerID

Releases(**artistID**, **dID**)

- artistID, dID -> artistID, dID

## Relational Schemas POST-NORMALIZATION

Artist_ContractedWith(artistID, username, email, age, userPassword, country, biography, numOfMembers, ranking, **labelID**)

FDs:

- artistID -> username, email, age, userPassword, country, biography, numOfMembers, ranking
- email -> artistID, username, age, userPassword, country, biography, numOfMembers, ranking

- The relation is already in BCNF
- Primary key: artistID
- Candidate keys: email
- Foreign key: labelID

MusicListener(listenerID, username, email, age, userPassword, favArtist, favSong, favGenre)

FDs:

- listenerID -> username, email, age, userPassword, favArtist, favSong, favGenre
- email -> listenerID, username, age, userPassword, favArtist, favSong, favGenre

- The relation is already in BCNF
- Primary key: listenerID
- Candidate keys: email
- Foreign key: N/A

RecordLabel(labelID, websiteURL, labelName)

FDs:

- labelID -> websiteURL, labelName
- websiteURL -> labelID, labelName

- The relation is already in BCNF
- Primary key: labelID
- Candidate key: websiteURL
- Foreign key: N/A

Normalized

Discography_Main(dID, discoName, genre, releaseDate, numOfLikes)

- Primary Key: dID
- Candidate Key: N/A
- Foreign Key: N/A

Discography_Ranking(<u>numOfLikes</u>, ranking)
- Primary Key: numOfLikes
- Candidate Key: N/A
- Foreign Key: N/A

**Normalization Steps:**
Original schema:
Discography(<u>dID</u>, discoName, genre, releaseDate, numOfLikes, ranking)
FDs:
- dID -> discoName, genre, releaseDate, numOfLikes, ranking
- numOfLikes -> ranking

- dID is the primary key, and the first FD fulfills the requirement for both 3NF and BCNF. However, the second FD violates both 3NF and BCNF. Therefore, normalization is needed.

- We will decompose the Discography table to be in BCNF.
- Closures:
  - {dID}+ = {dID, discoName, genre, releaseDate, numOfLikes, ranking)
  - {numOfLikes}+ = {numOfLikes, ranking}
- Non-trivial explicit & implicit FD's:
  - dID -> discoName
  - dID -> genre
  - dID -> releaseDate
  - dID -> numOfLikes
  - dID -> ranking
  - numOfLikes -> ranking
- **FOR THE PIC**: We will abbreviate each attribute as follows: dID = ID, discoName = n, genre = g, releaseDate = rD, numOfLikes = nL, ranking = r.
  We will abbreviate Discography into D.

D(ID, n, g, rD, nL, r)

Explicit FDs

ID → n, g, rD, nL, r

nL → r

Closures

$\{ID\}^+ = \{ID, n, g, rD, nL, r\}$

$\{nL\}^+ = \{nL, r\}$

Non-trivial Implicit (and Explicit) FDs from Closures

ID → n  nL → r

ID → g

ID → rD

ID → nL

ID → r

Decomposition

① Decompose D on nL → r  ID, n, g, rD, nL, r

D₁(nL, r)  D₂(nL, ID, n, g, rD)

Final decomposition : D₁(nL, r), D₂(ID, nL, n, g, rD)

FINAL DECOMPOSITION:

- Discography_Main(dID, discoName, genre, releaseDate, numOfLikes)
- Discography_Ranking(numOfLikes, ranking)

Album(**albumID**, numOfSongs, totalDuration)

FD:

- albumID -> numOfSongs, totalDuration

- The relation is already in BCNF
- Primary key: albumID
- Candidate key: N/A
- Foreign key: albumID

Song(**songID**, duration):

FD:

- songID -> duration

- The relation is already in BCNF
- Primary key: songID
- Candidate key: N/A
- Foreign key: songID

Playlist_Created(playlistID, **listenerID**, playlistName, numOfSongs, introduction, duration) :

FD:

- playlistID -> listenerID, playlistName, numOfSongs, introduction, duration

- The relation is already in BCNF
- Primary key: playlistID
- Candidate key: N/A
- Foreign key: listenerID

Merchandise_Sold(itemID, **artistID**, itemName, price):

FD:

- itemID -> artistID, itemName, price

- The relation is already in BCNF
- Primary key: itemID
- Candidate key: N/A
- Foreign key: artistID

Normalized

Venue_Main(venueID, country, postalCode, vName, capacity)

- Primary Key: venueID
- Candidate Key: N/A
- Foreign Key: N/A

Venue_LocationOne(country, postalCode, city)

- Primary Key: {country, postalCode}
- Candidate Key: N/A
- Foreign Key: N/A

Venue_LocationTwo(country, postalCode, province)

- Primary Key: {country, postalCode}
- Candidate Key: N/A
- Foreign Key: N/A

**Normalization Steps:**

Original Schema:

Venue(<u>venueID</u>, country, postalCode, city, province, vName, capacity)

FDs:

- venueID -> country, postalCode, city, province, vName, capacity
- country, postalCode -> city
- country, postalCode -> province

- venueID is the primary key, and the first FD identifies all the other attributes and fulfills the requirement for both 3NF and BCNF. However, the second and third FD's violate both 3NF and BCNF. Therefore, normalization is needed.

- We will decompose the table to be in BCNF.
- Closures:
  - {venueID}+ = {venueID, country, postalCode, city, province, vName, capacity}
  - {country, postalCode}+ = {country, postalCode, city, province}
- Non-trivial explicit & implicit FD's:
  - venueID -> country
  - venueID -> postalCode
  - venueID -> city
  - venueID -> province
  - venueID -> vName
  - venueID -> capacity
  - country, postalCode -> city
  - country, postalCode -> province
- **FOR THE PIC**: We will abbreviate Venue as V, and the attributes as follows: venueID = ID, country = co, city = ci, vName= n, capacity = ca, postalCode = pc, province = pr.

V( ID , co, ci , n, ca, pc, pr)

Explicit FDs

ID → co, ci, n, ca, pc, pr

co, pc → ci

co, pc → pr

Closures

{ID}+ = { ID, co, ci, n, ca, pc, pr}

{co, pc}+ = {co, pc, ci, pr}

Non-trivial Implicit (and Explicit) FDs from Closures

ID → co  ID → pr

ID → ci  co, pc → ci

ID → n  co, pc → pr

ID → ca

ID → pc

Decomposition

① Decompose V on co, pc → ci  ID, n, ca, pr  co, pc  ci

V1(co, pc, ci), V2 ( ID, n, ca, pr, co, pc)

② Decompose V2 on co, pc → pr  ID, n, ca  co, pc  pr

V3(co, pc, pr)  V4 (ID, n, ca, co, pc)

Final decomposition: V1(co, pc, ci), V3(co, pc, pr), V4(ID, n, ca, co, pc)

FINAL DECOMPOSITION:

- Venue_Main(venueID, country, postalCode, vName, capacity)
- Venue_LocationOne(country, postalCode, city)
- Venue_LocationTwo(country, postalCode, province)

LiveEvent_IsHeldAt(eventID, **venueID**, eventDate, startTime, duration):

FD:

- eventID -> venueID, eventDate, startTime, duration

- The relation is already in BCNF
- Primary key: eventID
- Candidate key: N/A
- Foreign key: venueID

IsIn(**albumID**, **songID**):

FD:

- albumID, songID -> albumID, songID

- The relation is already in BCNF
- Primary key: {albumID, songID}

- Candidate key: N/A
- Foreign key: albumID, songID

Includes(**<u>songID</u>**, **<u>playlistID</u>**):

FD:

- songID, playlistID -> songID, playlistID

- The relation is already in BCNF
- Primary key: {songID, playlistID}
- Candidate key: N/A
- Foreign key: songID, playlistID

PerformsAt(**<u>artistID</u>**, **<u>eventID</u>**):

FD:

- artistID, eventID -> artistID, eventID

- The relation is already in BCNF
- Primary key: {artistID, eventID}
- Candidate key: N/A
- Foreign key: artistID, eventID

Purchases(**<u>listenerID</u>**, **<u>itemID</u>**):

FD:

- listenerID, itemID -> listenerID, itemID

- The relation is already in BCNF
- Primary key: {listenerID, itemID}
- Candidate key: N/A
- Foreign key: listenerID, itemID

Attends(**<u>eventID</u>**, **<u>listenerID</u>**):

FD:

- eventID, listenerID -> eventID, listenerID

- The relation is already in BCNF
- Primary key: {eventID, listenerID}
- Candidate key: N/A
- Foreign key: eventID, listenerID

Releases(**<u>artistID</u>**, **<u>dID</u>**)

FD:

- artistID, dID -> artistID, dID

- The relation is already in BCNF

- Primary Key: {artistID, dID}
- Candidate Key: None
- Foreign Key: artistID, dID

## SQL DDL

(Note to TA: 🙂 The data types here are valid with Microsoft SQL Server)

```
CREATE TABLE RecordLabel(
      labelID INTEGER PRIMARY KEY,
      websiteURL VARCHAR(200) UNIQUE,
      labelName VARCHAR(50) NOT NULL)

CREATE TABLE Artist_ContractedWith(
      artistID INTEGER PRIMARY KEY,
      username VARCHAR(30) NOT NULL,
      email VARCHAR(320) UNIQUE NOT NULL, -- apparently the longest email
someone can create is 320 characters long
      age INTEGER,
      userPassword VARCHAR(50) NOT NULL, --cannot use "password" in SQL
      country  VARCHAR(56), --apparently the longest country name is 56
characters
      biography VARCHAR(200),
      numOfMembers INTEGER NOT NULL,
      ranking INTEGER,
      labelID INTEGER,
      FOREIGN KEY (labelID) REFERENCES RecordLabel)

CREATE TABLE MusicListener(
      listenerID INTEGER PRIMARY KEY,
      username VARCHAR(30) NOT NULL,
      email VARCHAR(320) UNIQUE NOT NULL, -- apparently the longest email
someone can create is 320 characters long
      age INTEGER,
      userPassword VARCHAR(50) NOT NULL,
      favArtist VARCHAR(50),
      favSong VARCHAR(50),
      favGenre VARCHAR(50))

      -- NORMALIZED DISCOGRAPHY
CREATE TABLE Discography_Main(
      dID INTEGER PRIMARY KEY,
      discoName VARCHAR(50) NOT NULL,
      genre VARCHAR(50),
      releaseDate SMALLDATETIME NOT NULL
numOfLikes INTEGER NOT NULL)
```

```sql
CREATE TABLE Discography_Ranking(
      numOfLikes INTEGER PRIMARY KEY,
      ranking INTEGER)

CREATE TABLE Album(
      albumID INTEGER PRIMARY KEY,
      numOfSongs INTEGER NOT NULL,
      totalDuration TIME NOT NULL,
      FOREIGN KEY (albumID) REFERENCES Discography_Main(dID))

CREATE TABLE Song(
      songID INTEGER PRIMARY KEY,
      duration TIME NOT NULL,
      FOREIGN KEY (songID) REFERENCES Discography_Main(dID))

CREATE TABLE Playlist_Created(
      playlistID INTEGER PRIMARY KEY,
      listenerID INTEGER NOT NULL,
      playlistName VARCHAR(50) NOT NULL,
      numOfSongs INTEGER NOT NULL,
      introduction VARCHAR(100),
      duration TIME NOT NULL,
      FOREIGN KEY (listenerID) REFERENCES MusicListener)

CREATE TABLE Merchandise_Sold(
      itemID INTEGER PRIMARY KEY,
      artistID INTEGER NOT NULL,
      itemName VARCHAR(50) NOT NULL,
      price DECIMAL NOT NULL,
      FOREIGN KEY (artistID) REFERENCES Artist_ContractedWith)

-- NORMALIZED VENUE

CREATE TABLE Venue_Main(
      venueID INTEGER PRIMARY KEY,
      country VARCHAR(56) NOT NULL,
      postalCode VARCHAR(7) NOT NULL,
      vName VARCHAR(50) NOT NULL,
      capacity INTEGER)

CREATE TABLE Venue_LocationOne(
      country VARCHAR(56) NOT NULL,
      postalCode VARCHAR(7) NOT NULL,
```

```sql
      city VARCHAR(90),
      PRIMARY KEY (country, postalCode))

CREATE TABLE Venue_LocationTwo(
      country VARCHAR(56) NOT NULL,
      postalCode VARCHAR(7) NOT NULL,
      province VARCHAR(90),
      PRIMARY KEY (country, postalCode))


CREATE TABLE LiveEvent_IsHeldAt(
      eventID INTEGER PRIMARY KEY,
      venueID INTEGER NOT NULL,
      eventDate DATE NOT NULL,
      startTime TIME NOT NULL,
      duration TIME,
      FOREIGN KEY (venueID) REFERENCES Venue_Main)

CREATE TABLE IsIn(
      albumID INTEGER,
      songID INTEGER,
      FOREIGN KEY (albumID) REFERENCES Album,
      FOREIGN KEY (songID) REFERENCES Song,
      PRIMARY KEY (albumID, songID))

CREATE TABLE Includes(
      songID INTEGER,
      playlistID INTEGER,
      FOREIGN KEY (songID) REFERENCES Song,
      FOREIGN KEY (playlistID) REFERENCES Playlist_Created,
      PRIMARY KEY (songID, playlistID))
```

```sql
CREATE TABLE PerformsAt(
      artistID INTEGER,
      eventID INTEGER,
      FOREIGN KEY (artistID) REFERENCES Artist_ContractedWith,
      FOREIGN KEY (eventID) REFERENCES LiveEvent_IsHeldAt,
      PRIMARY KEY (artistID, eventID))

CREATE TABLE Purchases(
      listenerID INTEGER,
      itemID INTEGER,
```

```
      FOREIGN KEY (listenerID) REFERENCES MusicListener,
      FOREIGN KEY (itemID) REFERENCES Merchandise_Sold,
      PRIMARY KEY (listenerID, itemID))

CREATE TABLE Attends(
      listenerID INTEGER,
      eventID INTEGER,
      FOREIGN KEY (listenerID) REFERENCES MusicListener,
      FOREIGN KEY (eventID) REFERENCES LiveEvent_IsHeldAt,
      PRIMARY KEY (listenerID, eventID))

CREATE TABLE Releases(
      artistID INTEGER,
      dID INTEGER,
      FOREIGN KEY (artistID) REFERENCES Artist_ContractedWith,
      FOREIGN KEY (dID) REFERENCES Discography_Main,
      PRIMARY KEY (artistID, dID))
```

## Tables

**INSERT STATEMENTS:**

Artist_ContractedWith(artistID, username, email, age, userPassword, country, biography, numOfMembers, ranking, **labelID**)

```
INSERT INTO Artist_ContractedWith
VALUES(1, 'HEIZE', 'heize11@gmail.com', 31, 'heizepassword', 'South
Korea', NULL, 1, NULL,1)

INSERT INTO Artist_ContractedWith
VALUES (2, '(G)I-DLE', 'gidle@email.com', 5, 'gidlepassword', 'South
Korea', '5-member self-producing Kpop girl group', 5, NULL, 2)

INSERT INTO Artist_ContractedWith
VALUES(3, 'Ryuichi Sakamoto', 'ryuichi11@gmail.com', 71,
'ryuichipassword', 'Japan', NULL, 1, NULL, 3)

INSERT INTO Artist_ContractedWith
VALUES(4, 'Lil Gittu', 'lilgittu@gmail.com', 30, 'gittupassword',
'Canada', 'Raps about databases', 1, NULL, 4)

INSERT INTO Artist_ContractedWith
VALUES(5, 'Taylor Swift', 'taylorswift@umgstores.com', 33,
'tspassword', 'United States', 'American singer-songwriter', 1, NULL,
5)

INSERT INTO Artist_ContractedWith
VALUES(6, 'Harry Styles', 'info@caa.com', 29, 'hspassword', 'United
Kingdom', 'English singer and actor', 1, NULL, 6)
```

| | artistID | username | email | age | userPassword | country | biography | numOfMembers | ranking | labelID |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | HEIZE | heize11@gmail.com | 31 | heizepassword | South Korea | NULL | 1 | NULL | 1 |
| 2 | 2 | (G)I-DLE | gidle@email.com | 5 | gidlepassword | South Korea | 5-member self-producing Kpop girl group | 5 | NULL | 2 |
| 3 | 3 | Ryuichi Sakamoto | ryuichi11@gmail.com | 71 | ryuichipassword | Japan | NULL | 1 | NULL | 3 |
| 4 | 4 | Lil Gittu | lilgittu@gmail.com | 30 | gittupassword | Canada | Raps about databases | 1 | NULL | 4 |
| 5 | 5 | Taylor Swift | taylorswift@umgstores.com | 33 | tspassword | United States | American singer-songwriter | 1 | NULL | 5 |
| 6 | 6 | Harry Styles | info@caa.com | 29 | hspassword | United Kingdom | English singer and actor | 1 | NULL | 6 |

MusicListener(listenerID, username, email, age, userPassword, favArtist, favSong, favGenre)

```
INSERT INTO MusicListener
VALUES (7, ‘sunshine1’, 'sunshine1@gmail.com', 20,
'sunshinePassword', 'Adele', 'Rolling in the Deep', 'jazz')

INSERT INTO MusicListener
VALUES (8, 'cutieee', 'lilcutie@gmail.com', 28, 'cutiePassword',
'Lana Del Ray', 'Young and Beautiful', 'classic')

INSERT INTO MusicListener
VALUES (9, 'bookWorm', 'bookLover@hotmail.com', 18, 'bookPassword',
'Ed Sheeran', 'Perfect', 'pop')

INSERT INTO MusicListener
VALUES (10, 'happyListener', 'happyListener@hotmail.com', 25,
'happyPassword', 'Taylor Swift', 'Love Story', 'pop')

INSERT INTO MusicListener
VALUES (11, 'rockLover', 'iLoveRock@yahoo.com', 29, 'rockPassword',
"Freddie Mercury', 'Bohemian Rhapsody', 'rock')

INSERT INTO MusicListener
VALUES (12, 'rainyyy', 'rainy@yahoo.com', 34, 'rainyPassword',
"Billie Eilish', 'Happier Than Ever', 'classic')
```

| | listenerID | username | email | age | userPassword | favArtist | favSong | favGenre |
|---|---|---|---|---|---|---|---|---|
| 1 | 7 | sunshine1 | sunshine1@gmail.com | 20 | sunshinePassword | Adele | Rolling in the Deep | jazz |
| 2 | 8 | cutieee | lilcutie@gmail.com | 28 | cutiePassword | Lana Del Ray | Young and Beautiful | classic |
| 3 | 9 | bookWorm | bookLover@hotmail.com | 18 | bookPassword | Ed Sheeran | Perfect | pop |
| 4 | 10 | happyListener | happyListener@hotmail.com | 25 | happyPassword | Taylor Swift | Love Story | pop |
| 5 | 11 | rockLover | iLoveRock@yahoo.com | 29 | rockPassword | Freddie Me... | Bohemian Rhaps... | rock |
| 6 | 12 | rainyyy | rainy@yahoo.com | 34 | rainyPassword | Billie Eilish | Happier Than Ever | classic |

RecordLabel(labelID, websiteURL, labelName)

```
INSERT INTO RecordLabel
VALUES (1, 'https://www.pnation.com/', 'PNation')

INSERT INTO RecordLabel
```

```
VALUES (2, 'http://www.cubeent.co.kr/intro', 'Cube Entertainment')

INSERT INTO RecordLabel
VALUES (3, 'https://www.commmons.com/en/', 'Commmons')

INSERT INTO RecordLabel
VALUES (4, 'https://www.cpsc304.com/', 'Gittu Entertainment')

INSERT INTO RecordLabel
VALUES (5, 'https://www.republicrecords.com/', 'Republic Records')
```

```
INSERT INTO RecordLabel
VALUES (6, 'http://www.columbiarecords.com/', 'Columbia Records')
```

| | labelID | websiteURL | labelName |
|---|---|---|---|
| 1 | 1 | https://www.pnation.com/ | PNation |
| 2 | 2 | http://www.cubeent.co.kr/intro | Cube Entertainment |
| 3 | 3 | https://www.commmons.com/en/ | Commmons |
| 4 | 4 | https://www.cpsc304.com/ | Gittu Entertainment |
| 5 | 5 | https://www.republicrecords.com/ | Republic Records |
| 6 | 6 | http://www.columbiarecords.com/ | Columbia Records |

Discography_Main(dID, discoName, genre, releaseDate, numOfLikes)

```
INSERT INTO Discography_Main
VALUES (1, 'Goblin OST', 'Kpop', '01/21/2017 12:00', 2000000)

INSERT INTO Discography_Main
VALUES (2, 'I FEEL', 'Kpop', '03/15/2022/ 02:00', 2000001)

INSERT INTO Discography_Main
VALUES (3, 'Merry Christmas Mr.Lawrence', NULL, '05/01/2015 12:00',
2000002)

INSERT INTO Discography_Main
VALUES (4, 'GITTU RAP', 'Hip Hop', '08/05/2023 12:00', 2000003)
```

```sql
INSERT INTO Discography_Main
VALUES (5, '1996', 'Dance', '06/04/1996 12:00', 2000004)

INSERT INTO Discography_Main
VALUES (6, 'Round and Round', 'Kpop', '01/21/2017 12:00', 2000005)

INSERT INTO Discography_Main
VALUES (7, 'Rain', 'Dance', '06/04/1996 12:00', 2000006)

INSERT INTO Discography_Main
VALUES (8, 'The Gittu Album', 'Hip Hop', '07/27/2023 12:00', 2000007)

INSERT INTO Discography_Main
VALUES (9, ‘Speak Now’, 'country pop', '10/25/2010 0:00', 2000008)

INSERT INTO Discography_Main
VALUES (10, 'Reputation', 'pop', '11/10/2017 12:00', 2000009)
```

| | dID | discoName | genre | releaseDate | numOfLikes |
|---|---|---|---|---|---|
| 1 | 1 | Goblin OST | Kpop | 2017-01-21 12:00:00 | 2000000 |
| 2 | 2 | I FEEL | Kpop | 2022-03-15 02:00:00 | 2000001 |
| 3 | 3 | Merry Christmas Mr.Lawrence | NULL | 2015-05-01 12:00:00 | 2000002 |
| 4 | 4 | GITTU RAP | Hip Hop | 2023-08-05 12:00:00 | 2000003 |
| 5 | 5 | 1996 | Dance | 1996-06-04 12:00:00 | 2000004 |
| 6 | 6 | Round and Round | Kpop | 2017-01-21 12:00:00 | 2000005 |
| 7 | 7 | Rain | Dance | 1996-06-04 12:00:00 | 2000006 |
| 8 | 8 | The Gittu Album | Hip Hop | 2023-07-27 12:00:00 | 2000007 |
| 9 | 9 | Midnights | pop | 2022-10-21 00:00:00 | 2000008 |
| 10 | 10 | Reputation | pop | 2017-11-10 12:00:00 | 2000009 |

Discography_Ranking(<u>numOfLikes</u>, ranking)

```sql
INSERT INTO Discography_Ranking
     VALUES (2000009, 1)

INSERT INTO Discography_Ranking
     VALUES (2000008, 2)
```

```sql
INSERT INTO Discography_Ranking
     VALUES (2000007, 3)

INSERT INTO Discography_Ranking
     VALUES (2000006, 4)

INSERT INTO Discography_Ranking
     VALUES (2000005, 5)

INSERT INTO Discography_Ranking
     VALUES (2000004, 6)

INSERT INTO Discography_Ranking
     VALUES (2000003, 7)

INSERT INTO Discography_Ranking
     VALUES (2000002, 8)

INSERT INTO Discography_Ranking
     VALUES (2000001, 9)

INSERT INTO Discography_Ranking
     VALUES (2000000, 10)
```

| | numOfLikes | ranking |
|---|---|---|
| 1 | 2000000 | 10 |
| 2 | 2000001 | 9 |
| 3 | 2000002 | 8 |
| 4 | 2000003 | 7 |
| 5 | 2000004 | 6 |
| 6 | 2000005 | 5 |
| 7 | 2000006 | 4 |
| 8 | 2000007 | 3 |
| 9 | 2000008 | 2 |
| 10 | 2000009 | 1 |

Album(**albumID**, numOfSongs, totalDuration)

```sql
INSERT INTO Album -- Goblin OST
     VALUES (1, 15, '00:53:00')

INSERT INTO Album -- I FEEL
     VALUES (2, 5, '00:17:15')

INSERT INTO Album
     VALUES (3, 19, '00:40:36')

INSERT INTO Album -- 1996
     VALUES (5, 12, '00:52:27')

INSERT INTO Album -- The Gittu Album
     VALUES (8, 5, '00:30:04')
```

| | albumID | numOfSongs | totalDuration |
|---|---|---|---|
| 1 | 1 | 15 | 00:53:00.0000000 |
| 2 | 2 | 5 | 00:17:15.0000000 |
| 3 | 3 | 19 | 00:40:36.0000000 |
| 4 | 5 | 12 | 00:52:27.0000000 |
| 5 | 8 | 5 | 00:30:04.0000000 |

Song(**<u>songID</u>**, duration):

```sql
INSERT INTO Song
     VALUES (4, '00:03:16')

INSERT INTO Song
     VALUES (6, '00:01:09')

INSERT INTO Song
     VALUES (7, '00:01:41')

INSERT INTO Song
     VALUES (9, '00:02:37')

INSERT INTO Song
     VALUES (10, '00:02:03')
```

| | songID | duration |
|---|---|---|
| 1 | 4 | 00:03:16.0000000 |
| 2 | 6 | 00:01:09.0000000 |
| 3 | 7 | 00:01:41.0000000 |
| 4 | 9 | 00:02:37.0000000 |
| 5 | 10 | 00:02:03.0000000 |

Playlist_Created(playlistID, **listenerID**, playlistName, numOfSongs, introduction, duration)

```
INSERT INTO Playlist_Created
     VALUES (1, 7, 'ryuichiPlaylist', 10, NULL, '00:30:45')

INSERT INTO Playlist_Created
     VALUES (2, 8, 'OSTPlaylist', 3, 'random songs', '00:09:03')

INSERT INTO Playlist_Created
     VALUES (3, 9, 'playlist3', 30, NULL, '01:40:39')

INSERT INTO Playlist_Created
     VALUES (4, 10, 'playlist4', 40, NULL, '02:18:43')

INSERT INTO Playlist_Created
     VALUES (5, 11, 'playlist5', 20, NULL, '01:08:03')
```

| | playlistID | listenerID | playlistName | numOfSongs | introduction | duration |
|---|---|---|---|---|---|---|
| 1 | 1 | 7 | ryuichiPlaylist | 10 | NULL | 00:30:45.0000000 |
| 2 | 2 | 8 | OSTPlaylist | 3 | random songs | 00:09:03.0000000 |
| 3 | 3 | 9 | playlist3 | 30 | NULL | 01:40:39.0000000 |
| 4 | 4 | 10 | playlist4 | 40 | NULL | 02:18:43.0000000 |
| 5 | 5 | 11 | playlist5 | 20 | NULL | 01:08:03.0000000 |

Merchandise_Sold(itemID, **artistID**, itemName, price):

```
INSERT INTO Merchandise_Sold
     VALUES (1, 1, 'Gold Prize Shirt', 70.0)

INSERT INTO Merchandise_Sold
     VALUES (2, 2, 'Neverland Lightstick', 20.0)
```

```sql
INSERT INTO Merchandise_Sold
     VALUES (3, 3, 'Cute Cup', 30.0)

INSERT INTO Merchandise_Sold
     VALUES (4, 4, 'Cool Hoodie', 40.0)

INSERT INTO Merchandise_Sold
     VALUES (5, 5, 'Collectable Ticket', 50.0)
```

| | itemID | artistID | itemName | price |
|---|---|---|---|---|
| 1 | 1 | 1 | Gold Prize Shirt | 70 |
| 2 | 2 | 2 | Neverland Lightstick | 20 |
| 3 | 3 | 3 | Cute Cup | 30 |
| 4 | 4 | 4 | Cool Hoodie | 40 |
| 5 | 5 | 5 | Collectable Ticket | 50 |

Venue_Main(venueID, country, postalCode, vName, capacity)

```sql
INSERT INTO Venue_Main
     VALUES (1, 'Canada',  'V6J 1Z2',  'The Modern Vancouver', 100)

INSERT INTO Venue_Main
     VALUES (2, 'Canada',  'V6B 5N6',  'Queen Elizabeth Theatre',
220)

INSERT INTO Venue_Main
     VALUES (3, 'Canada',  'V6T 2L6',  'Thunderbird Arena', 5054)

INSERT INTO Venue_Main
     VALUES (4, 'Canada',  'V11 111',  'Non-existent Arena 1', 100)

INSERT INTO Venue_Main
     VALUES (5, 'Canada',  'V22 222',  'Non-existent Arena 2', 200)
```

| | venueID | country | postalCode | vName | capacity |
|---|---|---|---|---|---|
| 1 | 1 | Canada | V6J 1Z2 | The Modern Vancouver | 100 |
| 2 | 2 | Canada | V6B 5N6 | Queen Elizabeth Theatre | 220 |
| 3 | 3 | Canada | V6T 2L6 | Thunderbird Arena | 5054 |
| 4 | 4 | Canada | V11 111 | Non-existent Arena 1 | 100 |
| 5 | 5 | Canada | V22 222 | Non-existent Arena 2 | 200 |

Venue_LocationOne(country, postalCode, city)

```
INSERT INTO Venue_LocationOne
     VALUES ('Canada',  'V6J 1Z2', 'Vancouver')

INSERT INTO Venue_LocationOne
     VALUES ('Canada',  'V6B 5N6', 'Vancouver')

INSERT INTO Venue_LocationOne
     VALUES ('Canada',  'V6T 2L6', 'Vancouver')

INSERT INTO Venue_LocationOne
     VALUES ('Canada',  'V11 111', 'Toronto')

INSERT INTO Venue_LocationOne
     VALUES ('Canada',  'V22 222', 'Whitehorse')
```

| | country | postalCode | city |
|---|---|---|---|
| 1 | Canada | V11 111 | Toronto |
| 2 | Canada | V22 222 | Whitehorse |
| 3 | Canada | V6B 5N6 | Vancouver |
| 4 | Canada | V6J 1Z2 | Vancouver |
| 5 | Canada | V6T 2L6 | Vancouver |

Venue_LocationTwo(country, postalCode, province)

```
INSERT INTO Venue_LocationTwo
     VALUES ('Canada',  'V6J 1Z2', 'British Columbia')

INSERT INTO Venue_LocationTwo
     VALUES ('Canada',  'V6B 5N6', 'British Columbia')

INSERT INTO Venue_LocationTwo
     VALUES ('Canada',  'V6T 2L6', 'British Columbia')

INSERT INTO Venue_LocationTwo
     VALUES ('Canada',  'V11 111, 'Alberta')

INSERT INTO Venue_LocationTwo
     VALUES ('Canada',  'V22 222, 'Yukon')
```

| | country | postalCode | province |
|---|---|---|---|
| 1 | Canada | V11 111 | Alberta |
| 2 | Canada | V22 222 | Yukon |
| 3 | Canada | V6B 5N6 | British Columbia |
| 4 | Canada | V6J 1Z2 | British Columbia |
| 5 | Canada | V6T 2L6 | British Columbia |

LiveEvent_IsHeldAt(eventID, **venueID**, eventDate, startTime, duration):

```
INSERT INTO LiveEvent_IsHeldAt
     VALUES (1, 1, '2022-12-11', '12/11/2022 20:00', '02:00:00')

INSERT INTO LiveEvent_IsHeldAt
     VALUES (2, 2, '2021-11-10', '11/10/2021 21:00', '01:00:00')

INSERT INTO LiveEvent_IsHeldAt
     VALUES (3, 3, '2020-09-09', '09/09/2020 12:00', '03:00:00')

INSERT INTO LiveEvent_IsHeldAt
     VALUES (4, 4, '2019-10-10', '10/10/2019 16:00', '02:00:00')

INSERT INTO LiveEvent_IsHeldAt
     VALUES (5, 5, '2002-01-30', '01/30/2002 20:00', '03:00:00')
```

| | eventID | venueID | eventDate | startTime | duration |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 2022-12-11 | 20:00:00.0000000 | 02:00:00.0000000 |
| 2 | 2 | 2 | 2021-11-10 | 21:00:00.0000000 | 01:00:00.0000000 |
| 3 | 3 | 3 | 2020-09-09 | 12:00:00.0000000 | 03:00:00.0000000 |
| 4 | 4 | 4 | 2019-10-10 | 16:00:00.0000000 | 02:00:00.0000000 |
| 5 | 5 | 5 | 2002-01-30 | 20:00:00.0000000 | 03:00:00.0000000 |

IsIn(**albumID**, **songID**):

```
INSERT INTO IsIn
     VALUES (1, 6)
INSERT INTO IsIn
     VALUES (3, 7)
INSERT INTO IsIn
     VALUES (2, 4)
INSERT INTO IsIn
```

```
      VALUES (5, 9)
INSERT INTO IsIn
      VALUES (8, 10)
```

| | albumID | songID |
|---|---|---|
| 1 | 1 | 6 |
| 2 | 2 | 4 |
| 3 | 3 | 7 |
| 4 | 5 | 9 |
| 5 | 8 | 10 |

Includes(**songID**, **playlistID**):

```
INSERT INTO Includes
      VALUES (4, 1)
INSERT INTO Includes
      VALUES (6, 2)
INSERT INTO Includes
      VALUES (7, 3)
INSERT INTO Includes
      VALUES (9, 4)
INSERT INTO Includes
      VALUES (10, 5)
```

| | songID | playlistID |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 6 | 2 |
| 3 | 7 | 3 |
| 4 | 9 | 4 |
| 5 | 10 | 5 |

PerformsAt(**artistID**, **eventID**):

```
INSERT INTO PerformsAt
      VALUES (3, 1)
INSERT INTO PerformsAt
      VALUES (1, 2)
INSERT INTO PerformsAt
      VALUES (2, 3)
INSERT INTO PerformsAt
      VALUES (4, 4)
INSERT INTO PerformsAt
```

```
      VALUES (5, 5)
```

| | artistID | eventID |
|---|---|---|
| 1 | 1 | 2 |
| 2 | 2 | 3 |
| 3 | 3 | 1 |
| 4 | 4 | 4 |
| 5 | 5 | 5 |

Purchases(**listenerID**, **itemID**):

```
INSERT INTO Purchases
      VALUES (7, 1)
INSERT INTO Purchases
      VALUES (8, 2)
INSERT INTO Purchases
      VALUES (9, 3)
INSERT INTO Purchases
      VALUES (10, 4)
INSERT INTO Purchases
      VALUES (11, 5)
```

| | listenerID | itemID |
|---|---|---|
| 1 | 7 | 1 |
| 2 | 8 | 2 |
| 3 | 9 | 3 |
| 4 | 10 | 4 |
| 5 | 11 | 5 |

Attends(**listenerID, eventID**):

```
INSERT INTO Attends
      VALUES (7, 1)
INSERT INTO Attends
      VALUES (8, 2)
INSERT INTO Attends
      VALUES (9, 3)
INSERT INTO Attends
      VALUES (10, 4)
INSERT INTO Attends
      VALUES (11, 5)
```

| | listenerID | eventID |
|---|---|---|
| 1 | 7 | 1 |
| 2 | 8 | 2 |
| 3 | 9 | 3 |
| 4 | 10 | 4 |
| 5 | 11 | 5 |

Releases(**artistID**, **dID**)

```
INSERT INTO Releases
      VALUES (1, 1)
INSERT INTO Releases
      VALUES (2, 2)
INSERT INTO Releases
      VALUES (3, 3)
INSERT INTO Releases
      VALUES (4, 4)
INSERT INTO Releases
      VALUES (5, 5)
INSERT INTO Releases
      VALUES (1, 6)
INSERT INTO Releases
      VALUES (5, 7)
INSERT INTO Releases
      VALUES (2, 8)
INSERT INTO Releases
      VALUES (4, 9)
INSERT INTO Releases
      VALUES (3, 10)
```

| | artistID | dID |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 1 | 6 |
| 3 | 2 | 2 |
| 4 | 2 | 8 |
| 5 | 3 | 3 |
| 6 | 3 | 10 |
| 7 | 4 | 4 |
| 8 | 4 | 9 |