**Creating Relations: Relating Tables Through Keys**

**Instructions**

In this practice problem, you will create three tables with data associations between them.

You will write your queries in the relevant .sql files to get the tests to pass.

Only one query should be written per file. Carefully check your syntax if you encounter errors. In particular, the name of the tables are very important because the automated tests depend on them to work properly.

**Set up a database (optional)**

This setup is optional, but if you would like a way to debug your queries, you can setup a database and connect it to DBeaver from your local machine. You can then execute any queries you write for this assignment and verify their outputs in DBeaver on your local machine.

1. Set up a new instance called thinkful\_music\_events on ElephantSQL. The instructions for creating a new database instance can be found in the "Creating and deleting databases" checkpoint of the previous module.
2. Connect DBeaver to your database instance and rename the database connection to thinkful\_music\_events for easy reference. The instructions for connecting DBeaver to your instance can be found in the "Installing DBeaver" checkpoint of the previous module.
3. Now, you can execute your queries in DBeaver.
4. After creating all the tables described below, you can run the src/seed.sql script in DBeaver to seed your database. Then you can use the SELECT \* FROM <table\_name> command to retrieve the records from the tables and check that the tables were properly populated. As you're creating your queries for this assignment, you can execute your queries in DBeaver to see if they're retrieving the datasets properly.

**Tables**

**Artists**

Write a query to create a table called artists in src/artists.sql. It should have the following fields:

* A artist\_id field that serves as a primary key for the table.
* A artist\_name field with a varying character datatype and a character limit of 255 characters.
* A genre\_name field with a varying character datatype and a character limit of 100 characters.

**Songs**

Write a query to create a table called songs in src/songs.sql. It should have the following fields:

* A song\_id field that serves as a primary key for the table.
* A song\_name field with a varying character datatype, a character limit of 100 characters, and a default string value of no song title.
* A album\_name field with a varying character datatype, a character limit of 100 characters, and a default string value of no album title.
* Create a one-to-many relationship between the artists and songs tables (i.e., an artist can be associated with many songs). To do this, reference the artist\_id from the artists table as a foreign key called artist in the songs table. Set a constraint on this foreign key so that it cannot be null.

**Concerts**

Write a query to create a table called concerts in src/concerts.sql. It should have the following fields:

* A concert\_id field that serves as a primary key for the table.
* A concert\_name field with a varying character datatype, a character limit of 255 characters.
* A concert\_date field with a date datatype.

Create a many-to-many relationship between the artists and concerts tables (i.e., an artist can be scheduled to perform at various concerts, and a concert can have many artists performing in it). To do this, create a join table called artists\_concerts in src/artists\_concerts.sql with the following fields:

* A artist\_id foreign key field with integer datatype that references artist\_id from the artists table.
* A concert\_id foreign key field with integer datatype that references concert\_id from the concerts table.
* A scheduled\_start\_at field with a time datatype.
* A scheduled\_end\_at field with a time datatype.
* Create a composite key out of the artist\_id and concert\_id columns. Set this key as the primary key for the table.