**Creating Relations: Other Joins**

**Instructions**

In this practice problem, you will write join queries to retrieve specified data from the songs table and the artists table. The tables have the following structures:

**artists table**

| artist\_id | artist\_name | genre\_name |

**songs table**

| song\_id | song\_name | album\_name | artists (FK)|

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. Only use aliases for specified fields.

**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 on your local machine. You can then execute any queries you write for this assignment and verify their outputs in DBeaver on your local machine.

If you have already set up DBeaver on your local machine to connect to an ElephantSQL-hosted database called thinkful\_music\_events for the previous checkpoint assignment, then you can just use the same setup to test out your queries for this assignment. Otherwise, follow the steps below:

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.

**Existing files**

| **file path** | **description** |
| --- | --- |
| setup/ | The files in this folder create and populate the artist and songs tables. You do not need to modify any of these files. |
| src/get-all-artists-and-songs.sql | Code your solution to task #1 in this file |
| src/get-all-songs-and-artists.sql | Code your solution to task #2 in this file |
| src/get-all-songs-and-all-artists.sql | Code your solution to task #3 in this file |
| test/ | This folder has the tests your code will run against. You do not need to edit any of these files. |

**Tasks**

**Outer Join Queries**

For each of the following files, create a query as described.

1. src/get-all-artists-and-songs.sql: Write a join query to return a list of ALL artists and their songs. Include the artist name, song title, and album name columns only in your result.
2. src/get-all-songs-and-artists.sql: Write a join query to return a list of ALL songs and their artists. Include the song name, album name, and artist name columns only in your result.
3. src/get-all-songs-and-all-artists.sql: Write a join query to return a list of ALL artists and ALL songs stored in the database. Include the artist name, song name, and album name columns only in your result. Alias the artist\_name and album\_name columns to artist and album respectively.