University of Toronto
CSC343, Fall 2015

Assignment 2

Due: November 13, at 10:00 pm sharp!

General Instructions

1. Please read the handout thoroughly before you proceed. Failure to follow instructions can affect your grade.
2. We strongly encourage you to do all your development for this assignment on CDF, either in the labs or via a remote connection.
3. Download the starter files from CDF:

   ```
   cp  ~csc343h/public_html/fall/assignments/a2/a2.zip  your_local_folder
   ```

   This archive includes:

   - The database schema, `testdata/artistdb.ddl`
   - A test dataset (see files in tablesfolder/ in a2.zip; these data files get loaded automatically into the corresponding tables by running the ddl file)
   - Expected answers for some of the queries in part 1, for the test dataset and schema provided.
4. Download the starter code for the Java portion of the assignment, `Assignment2.java`:

   ```
   cp  ~csc343h/public_html/fall/assignments/a2/Assignment2.java  your_local_folder
   ```

You are allowed, and in fact encouraged, to work with a partner for this assignment. You must declare your team (whether it is a team of one or of two students) and hand in your work electronically using your MarkUs svn repository. Please make sure that you can commit in your repository in advance (try commiting an empty text file, to make sure you/your group is able to commit).

Once you have submitted your files, be sure to check that you have submitted the correct version; new or missing files will not be accepted after the due date, unless your group has grace tokens remaining.

Schema

In this assignment, we will work with a music industry database. What follows is a brief description of the meaning of the attributes in the schema; to fully understand this database, read these descriptions, and have a look over the actual schema definition found in `artistdb.ddl`.

- **Artist**(artist_id, name, birthdate, nationality)
  A tuple in this relation represents an artist. An artist is considered a musician, a band, or a songwriter, with a given *name*. Names are stored in a single text attribute - for example, 'Justin Timberlake', 'Chris Martin', 'Coldplay', 'Metallica', 'Guns "n Roses' (notice how apostrophes are represented using double quotes in SQL), etc. The birthdate for a band is the date when it was formed. A band's nationality is considered based on where it was formed, regardless of the nationalities of its members.
- **Role**(artist_id, role)
  A tuple in this relation represents whether an artist is a 'musician' (solo artist), 'band', or a 'songwriter' (no space!!). An artist can fulfill multiple roles.

- **WasInBand**(artist_id, band_id, start_year, end_year)
  A tuple in this relation represents whether an artist belonged to a band, and during which timeframe. An artist can appear in different bands at different timeframes; also, an artist can belong to the same band for several distinct (non-overlapping) timeframes.

- **Album**(album_id, title, artist_id, genre_id, year, sales)
  A tuple in this relation represents that album *album_id*, with title *title*, performed by *artist_id*, belongs to genre *genre_id*, and was launched sometime during *year*, with total revenue of *sales* amount of $. You can assume an album is only categorized as one (dominating) genre, even if there may be influences from other genres.

- **Genre**(genre_id, genre)
  A tuple in this relation represents that a musical genre (e.g., 'pop', 'rock', 'heavy metal', etc.).

- **Song**(song_id, title, songwriter_id)
  A tuple in this relation represents that the song titled *title* was written by the Artist with the ID *songwriter_id*. If a band's song was composed by multiple members, then the songwriter_id is considered the artist_id of the band.

- **BelongsToAlbum**(song_id, album_id, track_no)
  A tuple in this relation represents the fact that a song *song_id* is included in the album *album_id*. A song that is the product of a collaboration between two musicians is only included in one album (the main artist's album). A song can feature in more than one album by different artists, only if it is a cover song.

- **RecordLabel**(label_id, label_name, country)
  A tuple in this relation represents a record label, and the country it is based in.

- **ProducedBy**(album_id, label_id)
  A tuple in this relation represents that the album *album_id* was produced by *record_label*.

- **Collaboration**(song_id, artist1, artist2)
  A tuple in this relation represents a collaboration between artists with IDs *artist1* and *artist2* on song *song_id*. Artist1 is the 'main artist' (the only one of the two who has this song included in their album), while Artist2 is the guest artist (for example, for the song 'One' by U2 featuring Mary J. Blige, U2 would be the main artist, Mary J. Blige would be the guest artist). A collaboration refers strictly to musicians and bands, and not to songwriters (a songwriter's authorship of a song does not count in this relation, and is instead represented in the Song relation). An artist cannot collaborate with themselves.

The schema definition uses four types of integrity constraints:

- Every table has a primary key ("`PRIMARY KEY`").
- Some tables use foreign key constraints ("`REFERENCES`").
- Some tables define other keys ("`UNIQUE`").
- Some attributes must be present ("`NOT NULL`").

Your work on this assignment must work on *any* database instance (including ones with empty tables) that satisfies these integrity constraints, so make sure you read and understand them.

Warmup: Getting to know the schema

To get familiar with the schema, ask yourself questions like these (but don't hand in your answers):

- Can two different artists have the same name in the **Artist** table?
  Yes, as long as the artist_id is different. Although copyright rules generally prevent this from happening, so we won't be testing such cases.

- Can there be two different albums with the same title released in different years? In the same year?
  Yes and yes. In fact, the same artist could (due to lack of inspiration perhaps) name two of their own albums using the same title. The album_id will be different though.
- Can an artist work on an album both as a musician and a song writer? *Yes, absolutely.*
- If several members of a band write a song, the songwriter is considered the band. This means that once a member leaves a band, he/she can do a cover of basically their own song. In this case, which one is the songwriter_id for this cover song? What about the album_id for this song? *In this case, the songwriter_id will be the original songwriter - the band. The album_id will be the new solo album of the artist that left the band.*
- Can someone not be signed for a record label? What does that mean in the above schema? *That means that the artist/band is either an indie artist, or an independent song writer.*

Part 1: SQL Statements

In this section, you will write SQL statements to perform queries and changes to the recording artist database. Write your SQL statement(s) for each question in separate files named `q1.sql`, `q2.sql`, ...etc., and submit each file on MarkUs. You are encouraged to use views to make your queries more readable. However, each file should be entirely self-contained, and not depend on any other files; each will be run separately on a fresh database instance, and so (for example) any views you create in `q1.sql` will not be accessible in `q5.sql`. In fact, you should drop any views you create, at the end of each sql file. **Each of your files must begin with the line** `SET search_path TO artistdb;` Failure to do so will cause your query to raise an error, leading you to get a 0 for that question.

For questions which ask you to make a query ("Report ..."), your output must exactly match the specifications in the question: attribute names and order, and the order of the tuples.

We will be testing your code in the **CDF environment** using PostgreSQL. It is your responsibility to make sure your code runs on CDF before the deadline! **Code that works on your machine but not on CDF will not receive credit**.

---

1. **Born to be wild.** Report all artists and their nationalities who were born in the same year as the release of the first album by Steppenwolf. Your query should return an empty table if no such artist is found. To extract a year from a timestamp type, use the Extract function, e.g.: Extract (year from birthdate)

Attribute	
name	Name of artist
nationality	Nationality of artist
Order by	Name ascending
Duplicates?	No duplicates

2. **It takes two flints to make a fire.** Write a query to determine if there is a correlation between artist collaborations and better album sales. Identify all artists whose average album sales for albums that they had guest collaborators on, is higher than any albums for which they did not have any guest collaborators.

Attribute	
artists	Artist names
avg_collab_sales	Average sales for albums with collaborators
Order by	Artist name ascending
Duplicates?	No duplicates

3. **Show me the money.** Report the total sales for each record label, for each year, ordered first alphabetically by label name, then by year in chronological order. For each record label, do not report years with 0 revenue, if no album was produced by this record label that year.

Attribute	
record_label	Name of record label
year	Year in which the record label did produce at least one album.
total_sales	Total_sales for this year.
Order by	Record label name ascending, then year ascending
Duplicates?	No duplicates

4. **The wind of change.** As artists evolve, their music evolves as well (for the better or worst, depending on who you ask). Similarly, song writers sometimes explore writing songs in a genre outside of what one would expect. Write a query to report all artists who released albums belonging to at least 3 different genres, as well as song writers who wrote songs for albums categorized in at least 3 genres. An artist can appear in both capacities in the results, if they explored at least 3 genres in each capacity. Remember that a song is considered to be in the same genre as the album it is part of.

Attribute	
artist	Name of the artist
capacity	'Musician/band', or 'song writer'
genres	Number of genres that this artist explored
Order by	Musicians/bands first, then songwriters. Number of genres descending, artist name ascending.
Duplicates?	No duplicates

5. **Be yourself.** While many artists have dedicated song writers, some artists prefer to write their own music, to better express their beliefs, views, and personality. Write a query to find all the artists that released *at least one* album in which they *exclusively* wrote their own music.

Attribute	
artist_name	Name of the artist that writes their own music
album_name	Album name
Order by	Artist name ascending, then album name ascending
Duplicates?	No duplicates

6. **Due South.** Many artists or bands start out in their home country, then after gaining international success, decide to move in order to sign with large record labels. Find all Canadian artists (nationality is 'Canada') who released their first album as an indie artist (if multiple albums in their first year, at least one has to be indie), and later on got signed by a record label based in 'US' at any point in their career.

Attribute	
artist_name	Name of the Canadian indie artist signed later in their career by a US label
Order by	Artist name ascending
Duplicates?	No duplicates

7. **Cover me!** A lot of popular songs are brought back many decades later, through the interpretation of another musician or band, as cover songs. Find all the songs that have been covered at least once, and the names of the artists that produced a cover, including the original artist.

Attribute	
song_name	Name of the song being covered
year	Year when the song was first released or covered
artist_name	Name of the artist that covered the song (including original artist)
Order by	Song name ascending, then year ascending, then artist name ascending
Duplicates?	No duplicates

8. **Let's get the band back together!** Bands sometimes split on good terms when artists wish to go to solo careers. Other times tensions between members breaks up even the most successful band. Regardless, as time goes by, bands sometimes re-unite, even after decades of inactivity.

   Let's say that the band 'AC/DC' decided to get back together last year, in 2014, for one last hurrah. While the new album is still in production (and hence the title and songs are not yet known), we know for sure that its members are now yet again active in the band. Write a query to represent this in the current schema, by inserting new timeframes for *all* members of AC/DC, for the interval of 2014-2015.

9. **You don't have the moves like Jagger.** Quite often you hear of artists who leave a band due to clashing interests. In such cases, the band needs to find a replacement.

   Let's say that last year, in 2014, the band 'Maroon 5' has had enough of its lead singer 'Adam Levine', and decided to bring in 'Mick Jagger' from the 'Rolling Stones' as the lead singer. Write a query (or several queries) to update the database accordingly. Make sure you think carefully of all the changes that need to be made to adjust both bands properly. (You can assume that the database does not already contain albums released in 2015 for any of the two bands. You can also assume, for simplicity, that both Mick Jagger and Adam Levine only belonged to one band ever, without timeframe gaps).

10. **Some things you just can't 'unhear'..** The record music industry's Catchy Song Control Division (assume such an entity actually exists) decided that Michael Jackson's 'Thriller' album was so catchy that it could cause mass hysteria, and therefore it must be purged from the database. In fact, they decreed that any trace of the album ever having existed must *also* be removed from the database. Perform SQL commands to do the following:

- Remove the album ‘Thriller’ and all its songs.
- Remove extra information about the album: which record label produced it, collaborations on its songs.

Your commands should do nothing if the given artist is not found or no such album by the artist exists in the database.

Embedded SQL

For this part of the assignment, you will create a class in the file "Assignment2.java", which will allow you to issue queries using JDBC. We will use the tables defined above, as described in artistdb.ddl. If, for some reason, you feel you need to use an intermediate view to execute a query in a method, you must create it in that method. However, you must make sure to drop the view before exiting the method.

Before you begin, read these general instructions:

1. You may not use standard input or output. Doing so even once will result in the autotester terminating, causing you to receive a **zero** for this part.
2. Be sure to close all unused statements and result sets.
3. The database, username, and password must be passed as parameters, never "hard-coded". We will use the `connectDB()` and `disconnectDB()` methods to connect to the database with our own credentials. Our test code will use these to connect to a database. You should **not** call these in the other methods we ask you to implement; you can assume that they will be called before and after, respectively, any other method calls.
4. All of your code must be written in `Assignment2.java`. This is the only file you may submit for this part.
5. You may not change the method signature of any of the functions we've asked you to implement. However, you are encouraged to write helper functions to maintain good code quality.
6. As you saw in lecture, to run your code, you will need to include the JDBC driver in your class path. You may wish to review the related JDBC Exercise posted on the course website.

Open the starter code in `Assignment2.java`, and complete the following functions (we're providing the implementation of the first two functions for you). You must not change the names and signatures of any of these core functions.

1. `connectDB`: Connect to a database with the supplied credentials.
2. `disconnectDB`: Disconnect from the database.
3. `findArtistsInGenre`: Return a list of all musicians and bands who released at least one album in a given genre.
4. `findCollaborators`: Return a list of artists who collaborated with a given artist on some song. The list of artists should include both the case when an artist was a guest collaborator, as well as the main artist.
5. `findSongwriters`: Return a list of all the songwriters that wrote songs for a given artist. Do not include the artist themself, for artists that wrote their own music (either partially or exclusively).
6. `findCommonAcquaintances`: Return the common acquaintances for two given artists. A common acquaintance is considered a collaborator (either as a guest or main artist), or songwriter that worked with both given artists at any point during their careers.

For all queries, except the connectDB and disconnectDB, the result must be an ArrayList of artist names, as per the function signatures.

Note that the actual Java code does not need to be very complex for this assignment; we really just want to give you an opportunity to put together using a JDBC connector and SQL in a single program. To compile your code, don't forget to specify the classpath, including the JDBC driver, for example:

```
java -cp /local/packages/jdbc-postgresql/postgresql-8.4-701.jdbc4.jar:. Assignment2
```

You should have a basic knowledge of Java from CSC207, which is a prerequisite for this course. Nevertheless, if you have Java-specific questions, feel free to consult the documentation. For your reference, the main built-in Java class that you'll need to refresh is `ArrayList`, for which you can find documentation here: `http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`. You can find some additional material in section 3.2 of here: `http://www.vogella.com/tutorials/JavaCollections/article.html#javacollections_lists`.