CS377, Spring 2022

Assignment 2

Due: Sunday March 6, at 11:59PM ET sharp [Late submissions: see syllabus for policy]. Start early!

General Instructions

Ready for some SQL fun? This assignment will help you think of different ways of writing SQL queries and update statements that perform various interesting operations on a Music database. Note: You are encouraged to work on your *Google Cloud Platform* PostgreSQL instance for this assignment (use your free academic credit!).

Download the starter zip folder from the A2 directory on Canvas. You'll find the following:

- The database schema (`artistschema/artistdb.SQL`). Execute the lines in `artistdb.SQL` in your postgres instance. This script will create the schema and populate its tables from a public Google storage bucket.
- A test dataset (see files in `tablesfolder/`). A copy of these data files gets loaded automatically into the corresponding tables when you run the commands in `artistdb.SQL`; this is a local copy of the folder for your reference.
- Expected answers for some of the queries (in `expected-answers/`), based on the provided test dataset. Note that these answers were produced on a GCP PostgreSQL instance. There may be minor differences in the ordering of *strings*, if you run your queries on a different platform or operating system (i.e., on your laptop).
- Sample answer file (`q_sample.sql`); you **must** follow this style when writing your answers!

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 you have a late-token remaining and you resubmit within 24 hours of the deadline).

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.SQL`.

- **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. An *end_year* of 2020 indicates that the artist is still an active member of this band (you can assume this is an old database dump from the year 2020).
- **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 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.*

SQL Statements

After understanding the schema, you will now 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 Canvas. 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 match **exactly** the specifications in the question: attribute names and order, the order of the tuples, and how to treat duplicates. It is your responsibility to make sure your code runs with no errors and the output matches our expectations exactly.

---

1. **(12 points) Definitely not mainstream.** Which genre is the *least* popular, among all genres, in terms of the total sales generated by albums in it? For that least popular genre, find the musicians who have *only* performed in albums that are categorized in that genre. It is okay for your query (and other queries in this homework) to return an empty table if the current database instance does not have such musicians, but keep in mind that your queries should work on any valid database instance.

Attribute	
musician	Name of musicians who only performed in the least popular genre
genre	Name of least popular genre
Order by	Musician name ascending
Duplicates?	No duplicates

2. **(12 points) 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

3. **(12 points) 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

4. **(14 points) 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

5. **(14 points) 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 'America' at any point in their career.

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

6. **(14 points) 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

7. **(11 points) Quid pro quo, eh?** Justin Bieber is suing the artist who wrote and performed the song titled 'Close Your Eyes', claiming that they both had actually collaborated on that song. More specifically, Bieber claims that he should be officially acknowledged as a guest artist on that song, even though there are no public records verifying the collaboration. To avoid bad publicity, the other artist agreed to settle by listing Bieber as a guest artist on their song, 'Close Your Eyes', but they had one condition: Justin Bieber must first list the other artist as a guest collaborator as well on one of Bieber's hit songs, 'Never Say Never'.

   It is your job now to make sure the database tables reflect these ridiculous settlement terms.

8. **(11 points) Some things you just can't 'unhear'..** For mysterious reasons, the record music industry's governing council decided that Michael Jackson's 'Thriller' album must be purged entirely 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.