CPSC 321: Homework 2 *Due: Monday, Sept. 22, 2025*

Instructions. This homework involves both a write-up and an SQL coding component. You must submit both parts to Canvas by the end of the day (11:59 PM) on the due date. Your write-up must be typed (not handwritten), organized, and clearly marked (i.e., with respect to which question and part is being answered). Additionally, include your first and last name, the course (CPSC-321, Fall 2025), and the assignment (HW-2) at the top of your write-up. Your write-up must be submitted as a single PDF file called `hw-2.pdf`. For the SQL coding part, you must complete the starter scripts (`hw2-1.sql` and `hw2-3.sql`), which will be provided via Piazza. Your scripts must include the information described below and also have the file header comments filled out. Failing to fill out the file header comments (and not following directions in general) will result in an overall reduction in points for the assignment.

1. Use SQL to implement the airport, airline, flight, and segment tables from HW-1. Your tables should have the following relation schemas, with the given *primary keys* as well as *foreign keys* and *additional constraints* as appropriate (not shown—you must determine what these should be). Each table you create must also have at *least* five rows and should demonstrate that your tables are defined correctly (e.g., with respect to foreign keys and additional constraints). The SQL statements to create and populate the tables must be placed in your `hw2-1.sql` script. It must be possible for the grader to completely recreate your tables by running the script from within PostgreSQL. You must also provide a screenshot in the write-up showing that the tables were successfully created (by running the script) as well as showing the contents of each table. Finally, include two `INSERT` statements per table that should fail if your constraints are correct. Show the error output in your screenshot but comment out the two lines in your submitted `hw2-1.sql` file.

> airport(<u>id</u>, name, city, state, elevation)
>
> airline(<u>code</u>, name, main_hub, yr_founded)
>
> flight(<u>airline, flight_number</u>, departure, arrival, flights_per_wk)
>
> segment(<u>airline, flight_number, segment_offset</u>, start_airport, end_airport)

2. Design relation schemas for storing the following information about music albums, tracks, songs, music groups, group members (musicians), music genres, and record labels. (Note you will implement your design using SQL in Question 3). Your relations must support the following requirements and constraints. For each relation define appropriate *primary key*, *foreign keys*, `NOT NULL` constraints, additional *uniqueness* constraints, and any other constraints (as needed) in your design. Your schema should not use any "surrogate" keys. To describe your design, create a schema diagram for it and include it in your homework write-up. Be sure to list `NOT NULL` constraints and

any other constraints in your schema diagram. Your schema diagram must be professional looking (e.g., neatly organized, no crossing lines, easy to follow layout, etc.). Note that a pasted picture taken on your phone of a hand-drawn diagram will not be considered as a professional-looking diagram.

- Each album has a title, the year it was recorded, the group that recorded the album, the album's tracks (i.e., the specific recording of songs), and the album's record label. While two albums can have the same title, you can assume that a group only has one album with a given title.
- Each music group has a name, assumed to be unique across groups, and the year the group was formed.
- Each music group can be associated with *zero or more* music genres. Each genre has a unique label, e.g., "jazz", "rock", "pop", "rap", and a longer description of the genre itself. For instance, the description of "indie pop" is "Combines guitar pop with DIY ethic in opposition to the style and tone of mainstream pop music."
- Each group can be influenced by *zero or more* other music groups. For example, the Beatles were influenced by the Byrds, the Everly Brothers, Chuck Berry, and many others.
- Each musician has a first name, a last name, an optional stage name, and a birth year (if publicly known). For simplicity, you can assume no two artists have the same first and last name.
- Music artists can be members of *zero or more* music groups. An artist is a member of a group within a certain range of years (e.g., from 1991 to 2000). It could be the case that the artist is still a member of a group (e.g., from 2007 to the present). While in general musicians may join, leave, and rejoin a music group, assume for this assignment that if a musician leaves and later rejoins, we will simply update the original membership's end-date instead of creating a new row.
- In cases where a record album is recorded under the name of an individual artist, assume that the album is recorded by a group whose name is the artists' name. In this way, both the artist and the other musicians involved in recording the music for the album are captured. The group would be named the same as the artist.
- A music track represents a recorded song. A song represents the abstract musical work (e.g., which can be copyrighted). A track is a particular recording of a song (e.g., akin to a song "instance"). Each music track is associated with one or more musicians (that contributed to the recording) along with the song it is a recording of. Albums consist of potentially many tracks. A particular track can also be included on multiple albums (e.g., for compilation albums or re-releases). Each track also has the year it was recorded along with a unique track identifier (as a number).

- A song has a title and the year it was written. In addition, each song is associated with the musicians that it was written by. For simplicity, you can assume that no two songs have the same title.

3. [10pts] Implement your design in Question 2 by creating the corresponding SQL tables with appropriate primary keys, foreign keys, and constraints. Populate each table with enough data to demonstrate that all keys and foreign keys are working correctly. Similar to Question 1, the SQL statements to create and populate your tables must be placed in `hw2-3.sql`. It must be possible for the grader to completely recreate your tables by running the script from within PostgreSQL. Include a screen shot in your write-up showing that your tables were created successfully and showing each table's contents. Finally, include two `INSERT` statements per table that should fail if your constraints are correct. Show the error output in your screenshot but comment out the two lines in your submitted `hw2-3.sql` file.