Do not write your name on this worksheet. This allows the grader to grade anonymously. Thanks!

1. Complete exercise 2.11 in your textbook; state and explain your answer below.

2. Complete exercise 2.12 in your textbook; state and explain your answer below.

   (a) part a

   (b) part b

3. Suppose that we have the following movie database schema, where for each relation, the underlined attributes jointly form the primary key:

   `Movie(`<u>`title`</u>`, `<u>`year`</u>`, length, fgenre, studioName, producerCertNum)`
   `StarsIn(`<u>`movieTitle`</u>`, `<u>`movieYear`</u>`, `<u>`starName`</u>`)`
   `MovieStar(`<u>`name`</u>`, address, gender, birthdate)`
   `MovieExec(name, address, `<u>`certNum`</u>`, netWorth)`
   `Studio(`<u>`name`</u>`, address, presidentCertNum)`

   (We're making up that every non-acting movie exec/producer/etc in the movie industry has a "certificate number," which is referred to in a number of the tables above.)

   Express each of the following constraints by suggesting a foreign key that, if put in place, would enforce that constraint. The answer for each of these problems should be of the form

   "The attribute(s) X in relation R should be a foreign key, referencing attribute(s) Y in relation S."

   If the specified constraint is not possible to be done as a foreign key constraint, explain why.

   (a) The producer of a movie must be someone mentioned in `MovieExec`.

   (b) A movie that appears in `StarsIn` must also appear in `Movie`.

   (c) A star appearing in `StarsIn` must also appear in `MovieStar`.

   (d) Every movie in the relation `Movie` must appear with at least one star in `StarsIn`.

(*Thanks to Ullman and Widom.*)

4. The MusicBrainz project (`https://musicbrainz.org`) is an open music encyclopedia that collects data about music: artists, titles, release dates, and so on. There is lots of detailed information about its database schema (`https://musicbrainz.org/doc/MusicBrainz_Database`). I was looking around at it because it seemed like it might be a fun example, and came across some interesting details they had to work their way through: specifically, (`https://musicbrainz.org/doc/Artist_Credits`) caught my attention. Read the artist credits information page, and then look at the entire database schema (`https://tinyurl.com/5rhejfp`). We haven't looked at these diagrams in detail yet, but you should be able to make some sense out of focusing on the relations `artist_credit`, `artist_credit_name`, and `artist`.

   There are three examples on the MusicBrainz Schema page (`https://tinyurl.com/d7zf6gz`)(scroll down to the "Artist Credit" section) that describe situations where the artist credit concept is relevant. Specifically, those examples are:

   - "Queen & David Bowie" – two artists ("Queen" and "David Bowie"), no name variations, joined with "&"
   - "Jean-Michel Jarre" – one artist ("Jean Michel Jarre"), name variation "Jean-Michel Jarre"
   - "Tracy W. Bush, Derek Duke, Jason Hayes and Glenn Stafford" – four artists, no name variations, joined with commas and an "and".

   Draw relation instances for the `artist_credit`, and `artist_credit_name` relations, based on the following `artist` relation (whose ids I made up). I only included the attributes necessary for this problem:

   Relation: artist

id	name
1	Queen
2	David Bowie
3	Jean Michel Jarre
4	Tracy W. Bush
5	Derek Duke
6	Jason Hayes
7	Glenn Stafford

   For the `artist_credit` relation, you only need to include the attributes `id`, `name`, and `artist_count`. You should include all attributes shown in the schema diagram for `artist_credit_name`. If the examples don't give you the information that you need to fill in some of the values, make something up.

   For the `join_phrase` attribute, you may need to sometimes have a value that means there is no join phrase. Write NULL in that case. In the pure relational model, there is no such thing as "no value," but essentially any real database system allows you to use NULL as a value.

   I've provided sample relations on the next page to fill in these values.

   (*You could likely accomplish this task by downloading the actual MusicBrainz database and looking at the raw data. Don't do that. The point of this exercise is to get you thinking about design considerations.*)

Relation: artist_credit

id	name	artist_count

Relation: artist_credit_name

artist_credit	position	artist	name	join_phrase