Introduction to Database Design
Homework 1

Björn Thór Jónsson

February 2019

1 Instructions

Write SQL commands to retrieve the information requested below from your PostgreSQL database. Then enter each query, as well as the numerical answer to each question, in LearnIT. The query should not return anything except the answer. As this is a learning exercise, you should take time to test variants and make the queries as simple as you possibly can. The queries should also be well and consistently formatted, and as readable as possible, as SQL queries are generally part of your code base.

While subqueries are fine, you should not produce a sequence of several queries that allow you to answer the question or use "magic constants" in your queries. Note that in many cases, some (incorrect) variants of some queries could return the same results as a correct query. The fact that a query returns the correct answer, thus does not necessarily mean that the query is correct. You must therefore take care to insert additional data to test your queries adequately. In short, the query must work for **any instance of the schema**.

In Part 1, you can use the rowcount from PostgreSQL to determine the numerical answer. In Part 2, however, all the queries ask you to count the rows that satisfy some criteria. You should always first create a query to return all the rows satisfying the given criteria. Once your query is ready, you can then turn it into a counting query, either by replacing the output columns with a `count` statement, or by enveloping the query with a counting query:

```
select count(*)
from (
    select ...
) tmp;
```

In an exam, you would also get directions along the following lines (but given that this is not an exam you need not worry about partial credit): A query that returns more information will not receive full points, even if the answer is part of the returned result. If you are unable to write a working query you can still submit your attempt, and it may be given partial points; in this case include a brief description of the problem with the query.

Database description

Part 1: Sports

In Part 1 you will work with the database `Sports`. To start working with the database, import/run `HW1-P1.sql` found in LearnIT using the PostgreSQL DBMS on your laptop. The database holds results for track and field athletes in a sports club. For simplicity there are only seven sports; a number of other simplifying assumptions are also made. The database has 254 athletes of both genders and more than 10K results from 200 competitions. The data is purely fictional and randomly generated, except that the places where competitions have been held actually exist.

```
Gender(gender,description)
People(ID,name,gender,height)
Sports(ID,name,record)
Competitions(ID,place,held)
Results(peopleID,competitionID,sportID,result)
```

Primary and foreign key attributes are those whose names include `ID` (and constraints are defined in the provided database). The attribute `gender` is the prinary key of the `Gender` relation. The next three relations have their first attribute as primary key. The last relation has a composite primary key consisting of the first three attributes. Note that the records represent national records so the best result in the database may not match the record. The meaning of other relations and attributes should be self-explanatory.s

Part 2: Games

In Part 2 you will work with the database `imdb`. To start working with the database, import/run `HW1-P2.sql` found in LearnIT using the PostgreSQL DBMS on your laptop. The database contains information on movies and people. Most of the data is actually part of the IMDb real database, but some information is made up. The following is the relevant part of the database schema (the relations `person` and `movie` relations have some additional attributes that are not used in the exam).

`person`(id, name, ...)
`movie`(id, title, year, ...)
`genre`(genre, category)
`movie_genre`(movieId, genre)
`role`(role)
`involved`(movieId, personId, role)

All relations have a defined primary key (all underlined attributes) except `movie_genre`. In particular, the last relation has a composite primary key consisting of all three attributes. Foreign keys are defined where appropriate. The `role` relation is a lookup table with a list of all possible roles. The meaning of other relations and attributes should be self-explanatory.

Part 1

In this part, you are given (a description of) the correct answer to the given query. You should write a query to produce the information requested, and then use the row count given by PostgreSQL to answer the quiz. Answer each of the following questions using a single SQL query on the Sports database:

(1) The number of athletes with some incomplete result registrations.

This query should return a single row with a single column, with the number 75.

(2) The ID and name of all athletes who have not participated in any competitions.

This query should return three rows: (249, Jens Hvidt), (250, Inge Lauridsen), and (251, Peter Laudruup).

(3) The ID and name of athletes who have competed in June of 2002 or hold a record in High Jump.

This query should return 53 rows with two columns each.

(4) The ID and name of all athletes who hold a record in some sport but have never competed in any other sport.

This query returns a single row: (119, Peter Jansen)

(5) For each sport, show the ID and name of the sport and the best performance over all athletes and competitions. This last column should be called maxres and should be formatted to show at least one digit before the decimal point and exactly two digits after the decimal point. The query result should be ordered by the sport ID.

The result should be as follows:
0, High Jump, 2.11
1, Long Jump, 6.78
2, Triple Jump, 13.15
3, Shot Put, 16.66
4, Pole Vault, 5.52
5, Javelin, 60.46s
6, Discus, 25.2

(6) Show the ID and name of athletes who hold a record in at least two sports, along with the total number of their record-setting or record-equaling results.

This query returns a single row: (25, Jens Jansen, 21).

(7) Show the ID, name, and height of athletes who have achieved the best result in each sport, along with the result, the name of the sport, and a column called record? which contains 1 if the result is a record or 0 if the result is not a record. Note that

for each sport there may be many athletes who share the best result and all should be in the result. If you can print yes and no instead of 1 and 0, all the better.

Note that some versions of this query may execute quite slowly, while others are much faster. Try to look for the fast version or, if you happen to initially make a fast version, try to look for a slower version, to make the comparison.

This query should return 39 rows, with 6 columns each. Two example rows are:
204, Inge Jensen, 1.7, 6.78, Long Jump, Yes
221, Jan Hansen, 1.75, 25.2, Discus, No

(8) The number of athletes who have competed in at least ten different places. Note that it does not matter how many sports or competitions they competed in, just how many places they have competed in.

This query should return the number 206.

(9) Show the ID and name of all athletes who hold a record in all sports.

In this dataset, there is no such athlete. You should add records to produce one or more such athletes to test your query.

(10) Show the ID, name, record and worst result of all sports that have at least one result from every place where a competition has ever been held.

This query should return information for five such sports:
0, High Jump, 2.11, 1.06
1, Long Jump, 6.78, 3.39
2, Triple Jump, 13.15, 6.58
3, Shot Put, 16.66, 8.34
4, Pole Vault, 5.52, 2.76

Part 2

In this part, you are given the correct answer of a very similar query, but not the correct answer for the query you should produce in the end. Start by finding the correct form for the given answer, and then convert the query to the requested answer. As outlined above, each query should return only a single numerical answer in this part. Answer each of the following questions using a single SQL query on the `imdb` database:

(11) The `person` relation contains 573 entries with a registered height greater than 190 centimetres. How many entries do not have a registered height?

(12) In the database, there are 365 movies where the average height of all people involved is less than 165 centimetres (ignoring people where the height is not registered). For how many movies is the average height of all people involved greater than 190 centimetres?

(13) The `movie_genre` relation does not have a primary key, which can lead to a movie having more than one entry with the same genre. How many movies in `movie_genre` have such duplicate entries?

(14) According to the information in the database, 476 different persons acted in movies directed by 'Francis Ford Coppola'. How many different persons acted in movies directed by 'Steven Spielberg'?

(15) Of all the movies produced in 2002, there are 12 that have no registered entry in `involved`. How many movies produced in 1999 have no registered entry in `involved`?

(16) In the database, the number of persons who have acted in exactly one movie that they self directed is 603. How many persons have acted in more than one movie that they self directed?

(17) Of all the movies produced in 2002, there are 282 that have entries registered in `involved` for *all* roles defined in the `roles` relation. How many movies produced in 1999 have entries registered in `involved` for all roles defined in the `roles` relation? *Note: This is a relational division query which must work for any schema; you can not use the fact that currently there are only 2 different roles.*

(18) The number of persons who have had some role in some movie from *all* genres from the category 'Newsworthy' is 156. How many persons have had some role in some movie from all genres from the category 'Lame'?