5DV119 Introduction to Database Management Spring 2015
Obligatory Exercise 2

Submission deadline to avoid lateness penalty: February 09, 2015 at 0800 (8am)

1 Problem Statement

Given is the following relational database schema:

Airport(Code, City, Country, Latitude, Longitude)

Flight(**Airline**, FlightNumber, **Origin**, **Destination**)

Schedule(**Airline, FlightNumber**, Date, DepartureTime, ArrivalTime)

Airline(Name, Abbreviation, HomeCountry, WebSite)

Ticket(Number, **Airline, FlightNumber, Date**, Cost)

In the above schema, note the following conventions:

- The primary key of each relation is underlined.
- Foreign keys are shown in **boldface**. Specifically:
  - The attribute **Airline** in the Flight relation is a foreign key which references the Airline relation.
  - The attributes **Origin** and **Destination** in the Flight relation are each foreign keys which reference the Airport relation.
  - The attribute pair (**Airline, FlightNumber**) in the Schedule relation is a foreign key which references the Flight relation.
  - The attribute triple (**Airline, FlightNumber, Date**) in the Ticket relation is a foreign key which references the Schedule relation.

Find a solution, in SQL, to each of the queries in Section 2 which works with both PostgreSQL and MySQL.

For queries 11 and 12 (which appeared on Obligatory Exercise 1 for PostgreSQL only), solutions for this exercise must work with both PostgreSQL and MySQL in order to receive any credit. Submit solutions for these two queries even if your submitted solutions for Obligatory Exercise 1 already worked for both PostgreSQL and MySQL; the grader will not follow a request to find the solution in your submission of Obligatory Exercise 1.

Unless stated specifically to the contrary, each solution must eliminate all duplicates from the result and must provide meaningful column names. Furthermore, to keep the solutions simple and consistent, the following two restrictions apply:

(i) The solution must consist of a single SQL directive. Creation and subsequent use of temporary tables is not permitted.

(ii) Subqueries involving SELECT are allowed only in the WHERE and HAVING fields of queries and subqueries, and in the SET fields of UPDATE commands.

2 Queries to Be Solved

1. Find the airport with the greatest latitude. In the case of a tie, list all such airports.

2. Find the names of those airlines which do not have flights which depart from an airport in Germany. (This includes airlines which do not have any flights at all.)

3. Find the names of those airlines which have a flight whose destination airport is the same as the origin airport of some flight of Lufthansa. (The query must use the string 'Lufthansa' and not the airline abbreviation LH.)

4. Find the names of those airports, all of whose departures are *international*, in the precise sense that the destination airport is in a different country than the airport of departure. Exclude airports with no departures.

5. Find the names of those airlines which, for every airport in Germany with latitude less than 54, except possibly BER, have a flight whose origin is that airport.

6. For each airport, find the total number of distinct destinations for flights which originate at that airport. List the airport code, city, and country, as well as the number of distinct destinations, and sort first by number of destinations and then by airport code, with the greatest number of airports first. Include even airports with no flights. (Hint: left outer join)

7. For each country found in the Airport relation of the database, find the maximum, minimum, and average latitude over all airports which are located in that country, as well as the total number of airports for each such country.

8. Find the code, city, and country of that airport which has the least *positive* number of distinct destinations for flights which originate at that airport. (That is, the airport must have at least one originating flight.) In case of a tie, list all such airports. Give the number of destinations for that airport as well.

9. Find the sum of the ticket costs for each carrier for flights departing in the month of November 2015. Report 0 for those airlines with no ticket sales and order from highest to lowest. (Hint: The SQL directive `ORDER BY n` will order the result based upon the nth column.)

10. Find the codes of those airports which are the origin for flights to at least three distinct airports in France.

11. Find the names of those airlines which have both a flight with origin the airport with code TXL and a flight with origin the airport with code SXF.

12. Find the codes of those airports located in Berlin, Germany which do not have any scheduled departures.

3 Submission Rules

1. SQL code which generates a test database for this exercise is available on the course Web page.

2. Unless stated specifically to the contrary, all solutions must be valid under both PostgreSQL and MySQL, using the database servers of the Department of Computing Science. This means the server `postgres` for PostgreSQL and the server `mysql` for MySQL. If you develop your solutions on other systems, make sure to test them on the database servers of the Department of Computing Science, as different versions of PostgreSQL and MySQL may support different features.

3. Solutions to Queries 11 and 12 must be submitted, even if your solutions to Queries 3 and 7 and of Obligatory Exercise 1 worked for both MySQL and PostgreSQL. The grader will not check your submission for Obligatory Exercise 1 in order to award credit for missing solutions to Queries 11 and 12.

4. Unless stated specifically to the contrary, all solutions must be valid under both PostgreSQL and MySQL, using the database servers of the Department of Computing Science. This means the server `postgres` for PostgreSQL and the server `mysql` for MySQL. If you develop your solutions on other systems, make sure to test them on the database servers of the Department of Computing Science, as different versions of PostgreSQL and MySQL may support different features.

5. The SQL code for the solutions of each of the ten queries must be placed in two text files, one which runs under PostgreSQL and the other which runs under MySQL. These files must be runnable as input source to both PostgreSQL (using the `\i` directive) and MySQL (using the `\.` directive), respectively. This means in particular that all comments must be in SQL-compatible format. The solutions must be in order, from Query 1 to Query 10. For each query, the following three items must be given, in order.

   (a) A comment which states the query, in English.

   (b) A line of the form `SELECT 'Query n';`, with `n` the number of the query.

   (c) The SQL solution to the query.

   As an example, for a fictitious Query 13:

```
-- Find the names of all airlines which are located
-- in Bohemia.
SELECT 'Query␣13';
```

```
SELECT Name FROM Airline WHERE (HomeCountry='Bohemia');
```

The purpose of the second line is to indicate the query number in the output when the file is run as input source.

6. Solutions must be uploaded using the submission system found at the following URL:
   `https://www8.cs.umu.se/kurser/5DV119/VT15/handin/`
   Note the following regarding this system:

   (a) The system will prompt for three items: a file `postgresql.sql`, a file `mysql.sql`, and a list of group members.

   (b) The corresponding files on your system are selected via a Web-based file browser. The names of the files on your system need not match those names; the uploaded files will be renamed automatically.

   (c) If your solutions for PostgreSQL and MySQL are identical (this will most likely be the case), then upload the same file twice, once for each system. If you solutions for PostgreSQL and MySQL differ, you must provide clear comments, in your SQL code, explaining why this was necessary.

   (d) The ID of the user who is uploading the submission is included automatically in the list of group members, and so need not be repeated (although it causes no harm to do so). However, the IDs of the other group members must be given with the first submission.

   (e) The ID of each group member **must** be the user ID of that person at `cs.umu.se`; that is, the `xxx` part of the e-mail address `xxx@cs.umu.se`.

   (f) All group members **must** be registered for the course; otherwise, the system will not accept the submission.

   (g) Once a submission is made, new group members may not be added for this assignment.

   (h) The system does some basic syntax checking of the submission. Thus, it is strongly recommended that the initial submission be made ahead of the deadline, so any problems in format may be fixed without the risk of a lateness penalty.

   (i) Resubmissions may be made at any time, but those submitted after the deadline will be flagged as late.

   (j) Whether or not a submission has been made for a particular student may be checked using the *Labres* system. A link will be provided on the course home page.

7. No report or file containing results is necessary, and paper submission is neither required nor possible. It suffices to submit the two solution files electronically.

8. Remember that a correct solution must work for all instances of the database, and not just for the test database provided.

9. This exercise may be done either individually or in a group consisting of no more than four (4) individuals.

10. Specific information on how to submit solutions for grading will be provided shortly. The details have not yet been finalized.

4 Further Notes

- Remember that there are point penalties for late submission. See the course syllabus.
- It is not allowed to copy the work of others. The submission must be the original work of the individual(s) in the working group.
- The grader reserves the right to interview members of the working group about the solution.
- So that solutions may be discussed in a class meeting, students and/or groups that are very late in preparing a solution may be required to solve an alternate problem to receive credit for this exercise.