DD1368 VT18 Introduction to Databases

LABORATION 1: SQL

RULES

This lab takes a lot of time and effort. Read the rules to make sure that your solutions are allowed, or you *will* have to redo them.

#1: The labs must be done in groups of exactly two people. No larger groups are allowed, and if you have extraordinary extenuating circumstances that force you to do the labs alone, you must obtain permission to do so from the course leader. Both students in a group must be able to present all of the lab for the group to pass.

#2: You must present correct and valid solutions to all the given problems to pass the lab.

#3: This is a SQL lab. No other programming languages, either embedded in the database or external to it, are allowed.

#4: You are not allowed to hard-code anything except that which has been explicitly given to you in the problems. In particular, this means that constructs like *limit 1* or similar artificial ways of reducing the output are forbidden.

#5: You must utilize nothing but a *single* top-level SELECT statement to answer every problem (although you are, of course, allowed to use any number of sub-selects required within that top level statement). You are specifically forbidden from referencing any temporary data structures like views or temporary tables from your solutions.

GENERAL HINT

The use of common table expressions (CTEs) is necessary only in one problem, but you will find that many become *much* easier to solve if you use them. You are strongly encouraged to learn how to use CTEs and to utilize them in your solutions frequently.

DATABASE TO USE

Find the database to use for this lab, Mondial, at
https://www.dbis.informatik.uni-goettingen.de/Mondial/
Install the Postgresql version of the schema and the input statements in your personal database on nestor2.

PROBLEMS

1. Generate a list of all countries that do not have any islands.

2. Generate the ratio between inland provinces (provinces not bordering any sea) to total number of provinces.

3. Generate a table of all the continents and the sum of the areas of all those lakes that contain at least one island for each continent. If a lake is in a country that is situated on several continents, the appropriate share of the lake area should be counted for each of those continents.

4. Generate a table with the two continents that will have the largest and the smallest population increase fifty years from now given current population and growth rates, and the future population to current population ratios for these two continents.

5. Generate the name of the organisation that is headquartered in Europe, has *International* in its name and has the largest number of European member countries.

6. Generate a table of city names and related airport names for all the cities that have at least 100,000 inhabitants, are situated in America and where the airport is elevated above 500 m.

7. Generate a table of countries and the ratio between their latest reported and earliest reported population figures, rounded to one decimal point, for those countries where this ratio is above 10, that is to say those that have grown at least 10-fold between earliest and latest population count.

8. Generate a table with the three (3) cities above 5,000,000 inhabitants that form the largest triangle between them, measured as the total length of all three triangle legs, and that total length. Your solution should be on the output form:

```
 Name 1       | Name 2      | Name 3       | TotDist
-------------------------------------------------------
 Bagginsville | Mordor City | Minas Tirith | 1234567.2
```

You are allowed to treat the world as a Mercator projection for purposes of calculating distances, that is, to use the distance formulas for a plane, but you must consider that the north/south edges and the east/west edges, respectively, meet and handle that. Any solution that counts two cities just on each side of the date line as a world apart, for instance, is wrong and will not be admitted. Your solution is allowed to contain duplicate rows of the same cities. **Hint 1**: Filter out the cities matching the condition *first!* **Hint 2:** Solve the simpler problem of calculating the two cities furthest apart under the above conditions first.

9. Generate a table that contains the rivers *Rhein*, *Nile* and *Amazonas*, and the longest total length that the river systems feeding into each of them contain (including their own respective length). You *must* calculate the respective river systems of tributary rivers recursively.