SQL-based definition of data structures and data modification

Part 1: Best Restaurant database

In this exercise, we will be using the "Best Restaurant" database again. It consists of five tables in the USERDEMO schema (see exercises of last week). The structure of the tables is provided as follows:

zip: zipcode, zipname

zoodcountrycat (food country categories): fcccode, fccdescription

person: persid, persname, persfirstname

bestrest (list of best restaurants): bestrestid, bestrestname, streetname, streetno, zipcode, fcccode

report (report and opinion for a best restaurant): bestrestid, persid, opiniontext, opiniondate

Think carefully whether you want to use GROUP BY, HAVING, and nested queries in your SQLs. Try to answer the following questions:

a) How many restaurants are there for each city (means, with the same zipname)? List the city names and the numbers of restaurants.

b) Which cities have more than one restaurant? List the city names, and the numbers of restaurants.

c) How many restaurants are there for each food country category? List the category names, and the numbers of restaurants.

d) Who has/have reported most? List their names, and the number of reports. (Please note there might be more than one user; For example, two persons might contribute equal number of reports).

e) Which restaurant(s) has/have been reported most often? List their names, and the number of reports.

f) Create a ranking list for all restaurants: Most voted restaurants on top. On equality, the restaurant with the most recent report should appear first.

Part 2: UZH database

In this exercise, you will implement a database schema for the management of teaching spaces for the courses at the University of Zurich. You will create the tables in your own schema. You will fill the tables by entering your own data as well as by importing data from an already existing table (userdemo.courses).

Exercise 1

Create a table course in your USRxx-schema (usrxx). Use the graphical tools or use SQL statements directly. Be careful about the definitions of primary keys and the correct choice of data types.

Table name:	*course*
Columns:	*courseno*, integer
	coursename, varchar(128), not null
Primary key:	name: *course_pk*, over column *courseno*

Exercise 2

Create the table *room*. Make sure that *roomno* is defined as a Primary Key. The attribute *roomdesc* should always have unique values, which means that it should be defined as unique.

Table:	*room*
Columns:	*roomno*, integer
	roomdesc, varchar(32) not null
	capacity, integer
Primary key:	Name: *room_pk*, over column *roomno*
Unique key:	Name: *room_uk*, over column *roomdesc*

Exercise 3

a) Insert the values for room Y25-H-86 with a capacity of 50 people. Set *roomno* to 1. Try this through the GUI (Graphical User Interface) as well as with pure SQL.

b) Insert values for room Y03-G-85. The capacity is so far unknown. Set *roomno* as 2.

Exercise 4

Data for the courses are already stored in the schema userdemo. Transfer them to your own table *course*. This can be done through

```
INSERT INTO usrxx.course (courseno, coursename)
(SELECT ........ FROM userdemo.course ..... );
```

Excercise 5

Set up a new table *roomrelation* to logically link the two tables (*course* and *room*).

Table:	*roomrelation*
Columns:	courseno, integer roomno, integer dateandtime, timestamp
Primary key:	Name: *roomrelation_pk*, over two columns *courseno*, *roomno*
Foreign keys:	Name: *roomrelation_room_fk*, with column *roomno* in both tables Name: *roomrelation_course_fk*, with column *courseno* in both tables

Excercise 6

Define the following relationship: „Spatial databases“ are taught on Friday in room Y25-H-86, 03.10.2016 at 10.15.

Tip 1: Convert a string explicitly to a time stamp datatype with the following function:
`to_timestamp('2016-10-03 10:15','YYYY-MM-DD HH24:MI')`

Tip 2: Add a single row to a table with following template:

```
INSERT INTO roomrelation (field1, field2, field3, field4)
VALUES (..., ..., ..., ...);
```

Excercise 7

a) Delete room entry Y03-G-85: **`DELETE FROM usrxx.room WHERE`** …

b) Delete room entry Y25-H-86: **`DELETE FROM usrxx.room WHERE`** … What happens and why?

Excercise 8

Delete the table room: **`DROP TABLE usrxx.room;`**. Why does this not work? What happens if you execute **`DROP TABLE usrxx.room CASCADE CONSTRAINTS;`**?