Exercise 10

Advanced SQL 1: Aggregates, joins and recursion

For this exercise sheet, please use the prepared SQL script `DB-Exercise10-KH-Testdata.sql` from the Community.

Task 1: Aggregate functions

a) Use the MIN aggregate function to output the best grade for every event of a lecturer.
b) Use the MAX aggregate function to output the worst grade given by every lecturer.
c) Use the COUNT aggregate function to output the number of students for every event of a lecturer.
d) Combine the COUNT and AVG aggregate functions to output the average number of students in the events of every lecturer.

Task 2: Joins

In this task, we are not interested in the projection list, so simply output all attributes with the help of `select *`. For each subsequent join used, indicate whether it is a natural, equi, theta or semi join.

a) Output a list of all events of a lecturer for which no student has registered yet.
b) Use a left join to create a list of all students which also contains the information of the respective events attended.
c) Convert the left join from the previous task into a right join.

Task 3: Recursion

In this task, we will be dealing with recursions. Since a recursion is structured differently in SQL to the common imperative programming languages, we will proceed step by step on a similar basis to the bus example in the lecture. First, look at the `Ancestors` table in the .sql file for this exercise. There, the father and mother are specified for every person.

a) Output a list with the name of a person in the first column and the name of a direct ancestor, i.e. father or mother, in the second column.
b) Then use the query you created earlier in a `with` and name it something like `FirstDegreeRelationships`. Then add second-degree and third-degree relationships.
c) From the first two tasks, develop the recursion start and the recursion step, and then use recursion to output a list containing all family relationships.
d) Extend the previously created recursion to include the degree of relationship. To do so, it may be helpful to initially expand the queries from the first two tasks. Then output all relationships and sort by name in ascending order and by degree of relationship in descending order.