SQL test 3 30.05.2023

For each of the tasks below, except for task 2, write the appropriate SQL commands. We expect the solution in the form of a file containing the **contents** of the SQL commands, not the output of the queries.

Syntactically incorrect queries will not be checked. Check your solution, for example, by using the command `\i file.sql`.

Send your solutions through the form at the address `https://dbserv.stud.ii/`. You can send the file multiple times; only the latest version will be checked. Do it as often as possible! All data on the computers will be erased after a restart. If you encounter any issues, please contact the course instructor before restarting the computer.

The required file format for the solution is as follows:

```
-- mabi-firstname-lastname
-- Task 1
<query>
-- Task 2
-- Written comment (commented out)
-- cd
-- ...
-- Task 3
<queries>
```

Load the file `offers-2.sql` into your database.

Remember that there is redundancy in the database: in the `offer` table, we have a foreign key `company_id`, and at the same time, we have a foreign key `company_branch_id` pointing to a table where the foreign key is `company_id`. Unfortunately, currently, this data does not match; in such a situation, we say that the *company IDs are inconsistent.*

Task 1 (3 points) Write a query that will correct the values of `company_id` in the `offer` table, so that the company IDs are consistent, i.e., each of them matches the `company_id` in the corresponding entry in the `company_branch` table.

The query should modify one row.

Task 2 (3 points) Consider the cases when company ID inconsistencies can occur. In the response to this task, describe all possible situations (in a comment), e.g., `-- inserting a row into the table ... with an ID such that...`
The reference solution discusses three basic scenarios. You don't have to worry about data deletion - foreign keys ensure consistency here.

Task 3 (9 points) Write appropriate triggers that will ensure that the company IDs are always consistent. Follow these rules:

1. If new data is inconsistent (e.g., we insert a new row into the `offer` table that leads to company ID inconsistency), such a query should be rejected, and the system should output an appropriate message.
2. If a user makes a change that updates `company_branch_id` but "forgets" about `company_id`, the update to `company_id` should be ensured as well.
3. If ensuring consistency requires changing another table, it is better to do it once for the entire query, regardless of how many rows it modifies (the query from the first task might come in handy).