CSE 111 – DATABASE SYSTEMS
Lab 7 (**15 points**)

In this lab session, you will write a database application in `Java` or `Python`. This is achieved by passing `SQL` statements to the database for execution and processing the returned query results in the application. This separation in logic is necessary because the database has relatively limited functionality – storing data and executing `SQL` statements – while `Java` or `Python` are general programming languages with extensive libraries that implement diverse functions. While we provide you skeleton code both for `Java` (`Lab_7.java`) and `Python` (`Lab_7.py`), you have the freedom to choose which programming language you use. However, you have to completely implement the application in one of the two languages.
We present the lab requirements for `Java`—the same applies to `Python`. You have to implement only the following methods/functions, which operate on the `TPC-H` database from the previous labs:

- `createTable` creates a table `warehouse` and adds it to the `TPC-H` database. Table `warehouse` stores data on the warehouses suppliers own and has the following schema/attributes:

  ```
  w_warehousekey decimal(9,0) not null,
  w_name char(100) not null,
  w_capacity decimal(6,0) not null,
  w_suppkey decimal(9,0) not null,
  w_nationkey decimal(2,0) not null
  ```

  `w_warehousekey` is the unique identifier of the warehouse and has a numeric value. `w_name` is the name of the warehouse, while `w_capacity` is its capacity. `w_suppkey` is the identifier of the supplier that owns the warehouse. A warehouse is owned by a single supplier, while a supplier can own multiple warehouses. `w_nationkey` is the identifier of the nation where the warehouse is located. `w_suppkey` and `w_nationkey` take values from the corresponding attributes in tables `supplier` and `nation`, respectively.

- `dropTable` drops/eliminates table `warehouse` together with all its data from the `TPC-H` database.

- `populateTable` populates table `warehouse` with tuples corresponding to every supplier in the database. Two warehouses are created for every supplier. The nations where these warehouses are located are those that have the largest number of lineitems supplied by the supplier that are ordered by customers from that nation. In case of equality, the nations are sorted in alphabetical order and the first two are selected. The name of a warehouse is obtained by concatenating the supplier name with "___" and with the name of the nation where the warehouse is located. In order to determine the capacity of a warehouse, you have to compute the total size of the parts (`p_size`) supplied by the supplier to the customers in a nation. Then, the warehouse capacity is taken as the double of the maximum total part size across all the nations. The two warehouses owned by a supplier have the same capacity. Finally, the `w_warehousekey` value is set as an increasing number that is unique across the tuples in the table.

- `Q1` displays the entire content of the `warehouse` table sorted on `w_warehousekey` by performing a `SQL` query. **(3 points)**

- `Q2` computes the number of warehouses and the total capacity for the warehouses in every nation. The result is sorted in decreasing order of the number of warehouses and of the capacity, then alphabetical order of the nation name. **(3 points)**

- `Q3` computes the suppliers that have a warehouse in a given nation taken as input parameter. The nation where warehouses are located is read from the input file `input/3.in`. `Q3` prints the name of the supplier, the nation of the supplier, and the name of the warehouse—sorted in alphabetical order by supplier name. **(3 points)**

- `Q4` finds the warehouses from a given region that have capacity larger than a given threshold. The region name and the minimum capacity are parameters stored in the file `input/4.in`. `Q4` prints the warehouse name and its capacity in decreasing order of the capacity. **(3 points)**
- `Q5` determines the total capacity of the warehouses belonging to suppliers from a given nation in every region. The suppliers' nation is a parameter stored in the file `input/5.in`. If there are no warehouses in a region, then value `0` is printed for that region. `Q5` prints the region and the capacity sorted alphabetically by region. **(3 points)**

In order to complete the lab you have to perform the following tasks:

1. Log in to your GitLab account.
2. Explore the folders and files in the Lab 7 repo.
3. Create a merge request for the `Instructions` issue. This is done from the `Issues` tab. The result of the merge request is a new branch that copies the files from `master`.
4. Clone the repo to your local machine or the remote lab machine. You can choose to directly clone the branch for the merge request, or the `master` and then checkout the merge request branch.
5. Write the `Java` code that implements the required functionality in the corresponding methods in file `Lab_7.java`. If you use `Python`, you edit the file `Lab_7.py`. This is the only file you have to edit. Moreover, you have to write code only in the methods/functions specified above.
6. You can check the correctness of your queries by executing the command `make run` in the terminal. You have to be in the main lab folder. The expected output is available in `results/x.res`, where `x` is the number of the query. The output produced by your code is available in `output/x.out`. They have to match exactly for every query, e.g., `1.res` has to match with `1.out`. For queries that require parameters, you can find their values in the files `input/x.in`.
7. Commit the changes to the code file and then push to the GitLab server.
8. Check the output of the pipeline under the `CI / CD` tab to see if your push has passed all the tests.

The score for the lab is assigned based on passing the test cases and the commit/push history. The instructor and the TAs have access to the GitLab repos.