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

In this Lab session you will get familiar with the Open DataBase Connectivity (ODBC) API to access a SQLite database from a high-level programming language, such as Java, C++, PHP, etc. Specifically, below it is shown how to set up a connection between Java and SQLite using the JDBC API.

To connect to a SQLite database in Java, the following are required:

1. Java JDK version 6 or later (`http://www.oracle.com/technetwork/java/javase/downloads/index.html`)
2. SQLite version 3.6 or later (`http://www.sqlite.org/`)
3. JDBC SQLite driver (`https://github.com/xerial/sqlite-jdbc`)

Create a simple Java class and use the following commands to connect and control the database from within Java:

- Create an instance of the class `Connection` and connect to the SQLite database. Notice, that the connection parameter has 3 fields separated by ":". The first is always `jdbc`; the second is the database server name, e.g., `sqlite`; the third is a path to your database file (in our case, this is the same folder as the main java class):

```
Connection conn = DriverManager.getConnection("jdbc:sqlite:test.db");
```

- The main object that you have to create to control the database is a statement:

```
Statement stat = conn.createStatement();
```

- With a statement object you can create tables and insert values in the existing table:

```
stat.executeUpdate("drop table if exists student;");
stat.executeUpdate("create table student (name, university);");
stat.executeUpdate("insert into student values ('John Smith', 'UC Merced');");
```

- You can get query results using the class `ResultSet` and cursors:

```
ResultSet rs = stat.executeQuery("select * from people;");
while (rs.next()) {
  System.out.println("name = " + rs.getString("name"));
  System.out.println("university = " + rs.getString("occupation"));
}
rs.close();
```

- At the end of the program, always disconnect from the database:

```
conn.close();
```

When you compile the code, make sure that `sqlite-jdbc-(VERSION).jar` is included in your Java classpath. A complete tutorial is available at `https://github.com/xerial/sqlite-jdbc`.

As a lab assignment you are required to create Java methods that perform the following actions:

1. Connect Java to your TPCH database from the previous labs.
2. Create a method that adds a new table `warehouse` with the following attributes:

```
w_warehousekey decimal(3,0) not null,
w_name char(25) not null,
w_supplierkey decimal(2,0) not null,
w_capacity decimal(6,2) not null,
w_address varchar(40) not null,
w_nationkey decimal(2,0) not null
```

3. Create a method that asks the user for a new entry in the `warehouse` table. `w_warehousekey` is generated automatically by the system and the user is required to provide information for `w_supplierkey` and `w_nationkey` as a supplier and nation name (the input is a string). The system finds the appropriate `s_supplierkey` and `n_nationkey` automatically and sets them in the new tuple. If one of them does not exist, an error is returned and the insertion is canceled. An example of a user entry is:

```
Name: Warehouse\#000000001
Supplier: Supplier\#000000005
Capacity: 100
Address: 10, Blumenstrasse, Berlin
Nation: Germany
```

4. Create methods that execute the following queries and output the results to the screen (one method for each query):
   - Find the supplier with the smallest number of warehouses.
   - Find the maximum warehouse capacity across all the suppliers.
   - List all the warehouses in `EUROPE` with capacity smaller than `X`, where `X` is taken as an input from the user.
   - For a supplier name given by the user, find whether all its warehouses are capable to fit all its products (see `ps_partsupp`).
   - For a nation given by the user, print all the warehouses in that country, in descending order of their capacity.
   - `Supplier#000000002` is acquired by `Supplier#000000001`. Update the `warehouse` table to reflect this change in ownership. The actual names of the suppliers are taken as input from the user, they are not constants.
5. Disconnect from the database.

The methods are controlled from the context menu that starts automatically at the beginning of the program. The menu has 10 items, one for each method described above.

Notice that, while you are expected to write the code in Java, you can use any other programming language (e.g., because you are using other language for your project). You can find more information on how to connect SQLite to other programming languages at: `http://www.sqlite.org/cvstrac/wiki?p=SqliteWrappers`.

Requirements. For full score, you are required to demo your application to the TAs during the lab session of **Week 10 October 29**. Moreover, you have to upload your single Java file with the code to CatCourses.