CNT 4714 – Project Three – Spring 2024

Title: "Project Three: A Two-Tier Client-Server Application Using MySQL and JDBC"
Points: 100 points
Due Date: Sunday March 10, 2024 by 11:59 pm (WebCourses Time)

Objectives: To develop two-tier Java based client-server applications interacting with a MySQL database utilizing JDBC for the connectivity. This project is designed to give you some experience using the various features of JDBC and its interaction with a MySQL DB Server environment.

Description: In this project you will develop a Java-based GUI front-end (client-side) application that will connect to your MySQL server via JDBC. The application will let clients with various permissions execute SQL commands against different databases. You will also develop a more specialized GUI interface as a monitoring application.

Detailed Description: You will ultimately develop two different Java applications that will allow clients (the end-users) to execute SQL commands against a remote database. The first of these applications will allow general types of end-users (clients that you will create) to issue SQL commands against various databases. The second of these applications will be very closely based on the first application but is restricted to a specialized accountant client (more on this application below). For the first application, you will create a Java GUI-based front-end that will accept any MySQL DDL or DML command, pass this through a JDBC connection to the MySQL database server, execute the statement and return the results to the clients. Note that while technically your application must be able to handle any DDL or DML command, we won't actually use all of the commands available in these sublanguages. For one thing, it would be quite rare to allow a client to create a database or a table within a database. Note too, that the only DML command that uses the `executeQuery()` method of JDBC is the Select command, all other DML and DDL commands utilize `executeUpdate()`. Some screen shots of what your Java GUI front-end should look like are shown below. Basically, this GUI is an extension of the GUI that was developed in the lecture notes and is available on WebCourses as DisplayQueryResults.java. Your Java application must give the user the ability to execute any SQL DDL or DML command for which the user has the correct permissions. User information for connections will be maintained in properties files, but each user must supply their username and password (for their MySQL server account) via the GUI for verification purposes (more later). You will be able to start multiple instances of your Java application and allow different clients to connect simultaneously to the MySQL DB sever, since the default number of connections is set at 151 (See your Workbench options file under the networking tab). In addition to the client interactions with your application, a background (business logic) transaction logging operation will occur which keeps a running total of the number of queries and the number of updates that have occurred via each different user's interactions with your application (aggregate over all separate connections per user). This is a separate database (i.e., a completely different database than any to which a client user can connect), to which the application will connect, using special application-levels privileges in a separate properties file. This separate properties file is not accessible by any end user. Each user operation will cause the application to make this connection and update the operational logging database table. More details on this aspect of the application are shown below and will be covered in the Q&A sessions. The second application that you develop will be essentially

a ”special-case” version of the first application, but will be restricted to an accountant-level client whose permissions are restricted to viewing (querying) this transaction logging database.

Once, you've created your main application, you will execute a sequence of DML and DDL commands and illustrate the output from each in your GUI for three different users. For this project you will create, in addition to the root user, two client users with limited permissions on the databases (see below), and an accountant-level users with limited permissions on the transaction logging database.. The root user is assumed to have all permissions on the databases, any command they issue will be executed. The client users will be far more restricted.

Restrictions:

1. Your source files for the main application should begin with comments containing the following information:

```
/*
        Name: <your name goes here>
        Course: CNT 4714 Spring 2024
        Assignment title: Project 3 – A Two-tier Client-Server Application
        Date:  March 10, 2024

        Class:  <name of class goes here>
*/
```

2. Your source files for the secondary application should begin with the comment containing the following information:

```
/*
Name: <your name goes here>
Course: CNT 4714 Spring 2024
Assignment title: Project 3 – A Specialized Accountant Application
Date:  March 10, 2024

Class:  <name of class goes here>
*/
```

3. Your applications must provide a user interface, similar to the ones shown below, that will allow the proper category of user the ability to connect to selected databases via properties files. Your applications (both of them) **must** verify that the user credentials (username and password) entered via the interface match with the user credentials found in the properties file that was selected via the interface. If the credentials do no match, then no connection is established.
4. Non-query commands should display a message to the user regarding the status of the executed command (see below).

References for this assignment:

Notes: Lecture Notes for MySQL and JDBC. Q&A Session Videos.

Input Specification:

The **first step** in this assignment is to login to the MySQL Workbench as the **root user** and execute/run the script to create and populate the backend database. This script is available on the project page and is named “`project3dbscript.sql`”. This script creates a database named

project3. You can use the MySQL Workbench for this step, or the command line whichever you prefer. You will also be using the **bikedb** from the notes. The script to create this database is already on Webcourses and if you have gone through the examples from the Module 3 notes (as you should have done), then you will have already created this database. If not, create it now.

The **second step** is to create a client-level user and assign authorizations for the user named `client1`. This client-level user is a new user with specific permissions. By default your root user has all permissions on the **project3** and **bikedb** databases. Use either SQL Grant statements from the command line or the MySQL Workbench (see separate document for details on how to accomplish this task) to set the specific permissions for the client as follows:

Register the new user named **client1** (assign them the password ***client1*** – ignore the MySQL warning on weak password setting) and assign to this user only select privileges on the **project3** and **bikedb** schemas.

The **third step** is to create another client-level user and assign authorizations for the user named `client2`. This client-level user is a new user with specific permissions. By default your root user has all permissions on the **project3** and **bikedb** databases. Use either SQL Grant statements from the command line or the MySQL Workbench (see separate document for details on how to accomplish this task) to set the specific permissions for the client as follows:

Register the new user named **client2** (assign them the password ***client2*** – ignore the MySQL warning on weak password setting) and assign to this user only select and update privileges on the **project3** and **bikedb** schemas.

The **fourth step** is to create the **operationslog** database using the `project3operationslog.sql` script. This script file is also available on WebCourses. This database will only be used and accessed by the project 3 application and the accountant-level user (see below), it is not intended to be accessed by normal end users. You will need to create a specific user for this database. We will call this user **project3app**. Both the username and password should be ***project3app*** for this user. The **project3app** user will need to be assigned select, insert, and update privileges on the **operationslog** database.

The **fifth step** is to create an account for the user named `theaccountant` and assign authorizations for this user only on the **operationslog** database. **Theaccountant** is aa special-case user who is maintaining user-level transaction profiles. **Theaccountant** user has only select privilege on the **operationslog** database. Use either SQL Grant statements from the command line or the MySQL Workbench (see separate document for details on how to accomplish this task) to set the specific permissions for **theaccountant**. As with the other two clients make the username and password the same for theaccountant.

Output Specification:

There are four parts for the output for this project.

1. Part 1 is to provide screen shots from your application which clearly show the complete query/command expression and results for each of the commands that appear in the script named: **`project3rootuserscript.sql`** available on the course website. There are

ten different commands in this script and some of the commands will have more than one output capture (see below).

2. Part 2 is to provide screen shots from your application which clearly show the complete query/command expression and results for each of the commands that appear in the script named: **`project3client1userscript.sql`** available on the course website. There are seven different commands in this script and some of the commands will have more than one output capture (see below).
3. Part 3 is to provide screen shots from your application which clearly show the complete query/command expression and results for each of the commands that appear in the script named: **`project3client2userscript.sql`** available on the course website. There are seven different commands in this script and some of the commands will have more than one output capture (see below).
4. Part 4 is to provide three screenshots. The first two are taken from the perspective the accountant specific interface application (see page 16 number 12 and page 18 number 14 for examples), that shows: (1) the results of the query –

   ```
   "select num_queries
   from operationscount
   where login_username = "root@localhost";
   ```

   and (2) `select * from operationscount;`

   The final screenshot is taken from the perspective of the Workbench (see page 17 number 13). Login to the MySQL Workbench as the root user and execute the following query:

   ```
   select * from operationscount;
   ```

To produce your final output, first recreate the database, then run the root user commands followed by the client commands in script order within each script file.

Deliverables:

1. All of the .java files associated with your project.
2. All 17 screenshots from the execution of the commands specified in the **`project3rootuserscript.sql`** script.
3. All 11 screenshots from the execution of the commands specified in the **`project3client1userscript.sql`** script.
4. All 11 screenshots from the execution of the commands specified in the **`project3client2userscript.sql`** script.
5. All 3 screenshots from item 4 above.
6. A screenshot showing a mismatch between the user-entered credentials and the selected properties file resulting in no connection to the database being established. See page 13 in this document.

All of the above deliverables should be uploaded to WebCourses no later than 11:59pm Sunday March 10, 2024. **Zip everything into a single archive to upload.** Be sure to clearly label each screen shot. Use the convention: RootCommand1, RootCommand2A, RootCommand2B, and so on. Similarly for Client1Command1, Client1Command2A, and so on.

Additional Details:
Beginning on page 5 are screen shots of the initial GUIs and subsequent uses of the UIs. Notice in the main application, that there are drop-down lists for selecting the various properties file that will be used to make the user connection. There will be properties files that maintain database driver and URL information as well as properties files that maintain user details for the connections. The driver and database URL will be maintained in properties files using the naming convention databasename.properties. The user credentials will be maintained in properties files using the naming convention username.properties. user credentials along with the JDBC driver and database URL will be specified in these files. The client must enter only their user credentials (username and password) through the GUI. Your application must verify that the user-entered credentials match those in the specified properties file before making a connection to the database. If the user entered credentials do not match those in the specified properties file, a message will be displayed to the user and no connection to the database will be established. Note that the properties file associated with the theaccountant user is not split into two properties files…all properties are contained in one file for this application only.

You must provide buttons for the user to clear the command window as well as the result window. You must provide buttons for establishing a connection as well as terminating a connection. The status of the connection should be returned to the GUI and displayed in the connection area. You must also provide buttons for clearing the command input area as well as the results area

The output of all SQL commands should be returned to the SQL Execution Result window. Please note that only single SQL commands can be executed via this application (we will not execute scripts of commands). We will also not go to the effort of making the application display the results of MySQL-specific commands. (When a MySQL-specific command is executed, the SQL Execution Result window does not need to display any results, if you wanted to you could display the line "MySQL command executed" in the results window, but this is not required.)

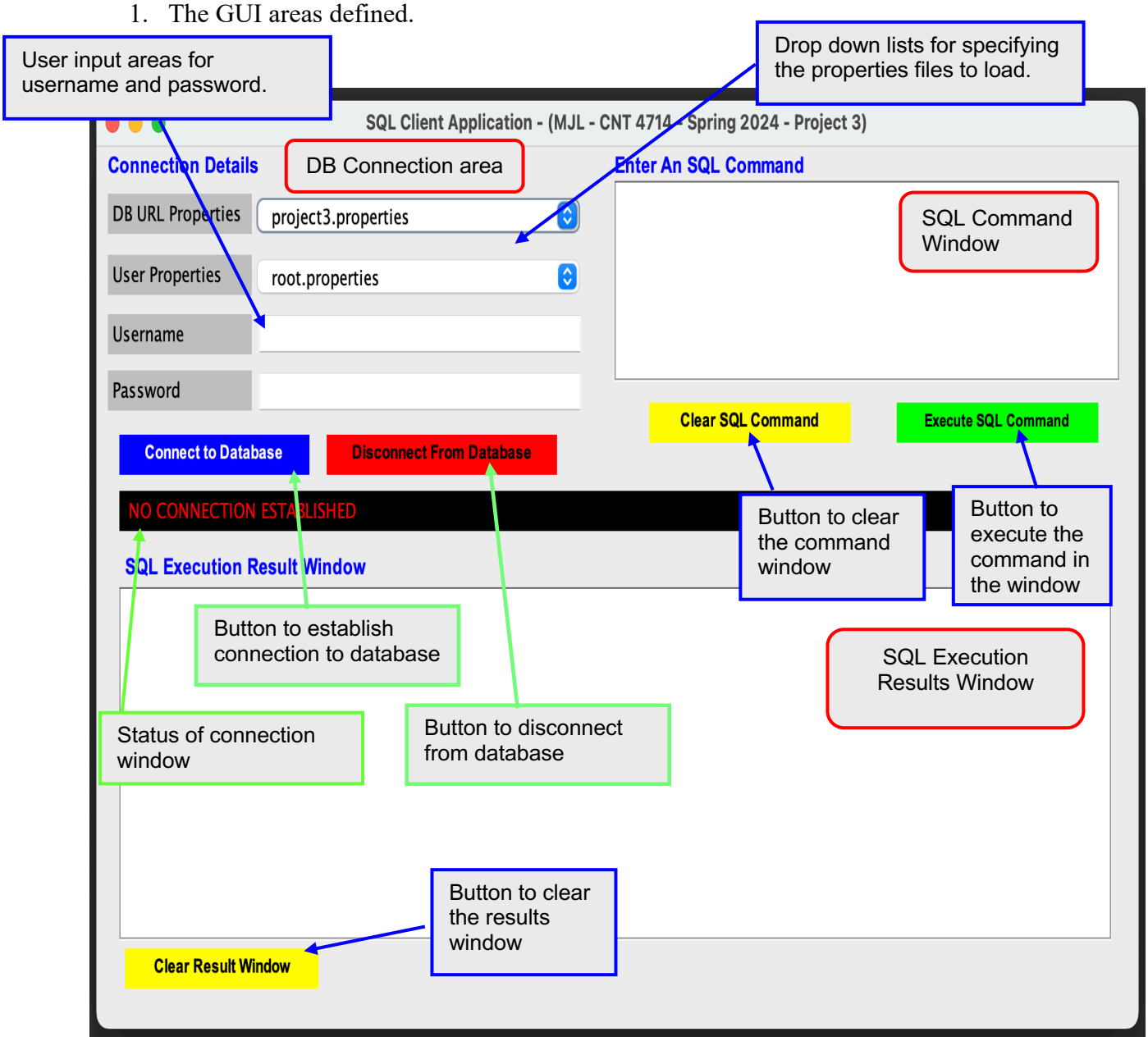
As each command in the various user scripts is executed (only successful commands – some of the commands in the various scripts will not be successful) the **operationscount** table in the **operationslog** database must be updated by your application. Note that the **operationscount** table is initially empty as defined in the creation script. Each user's queries and updates will be logged (counted) separately. Your application must obtain a connection to the **operationslog** database and perform the updates with project3app user credentials. Only successful operations will be logged – any transaction that errors will not increment any counter. These operations are invisible to the end user (regardless of who the user is, including root users). The application must connect to the **operationslog** database using a properties file which contains all necessary connection information. Note that any commands executed by theaccountant user will not be logged into this database.

Note that for non-query DML and DDL commands, before and after screen shots must be taken to illustrate the basic effect of the command. See pages 11-12 for an illustration of this.

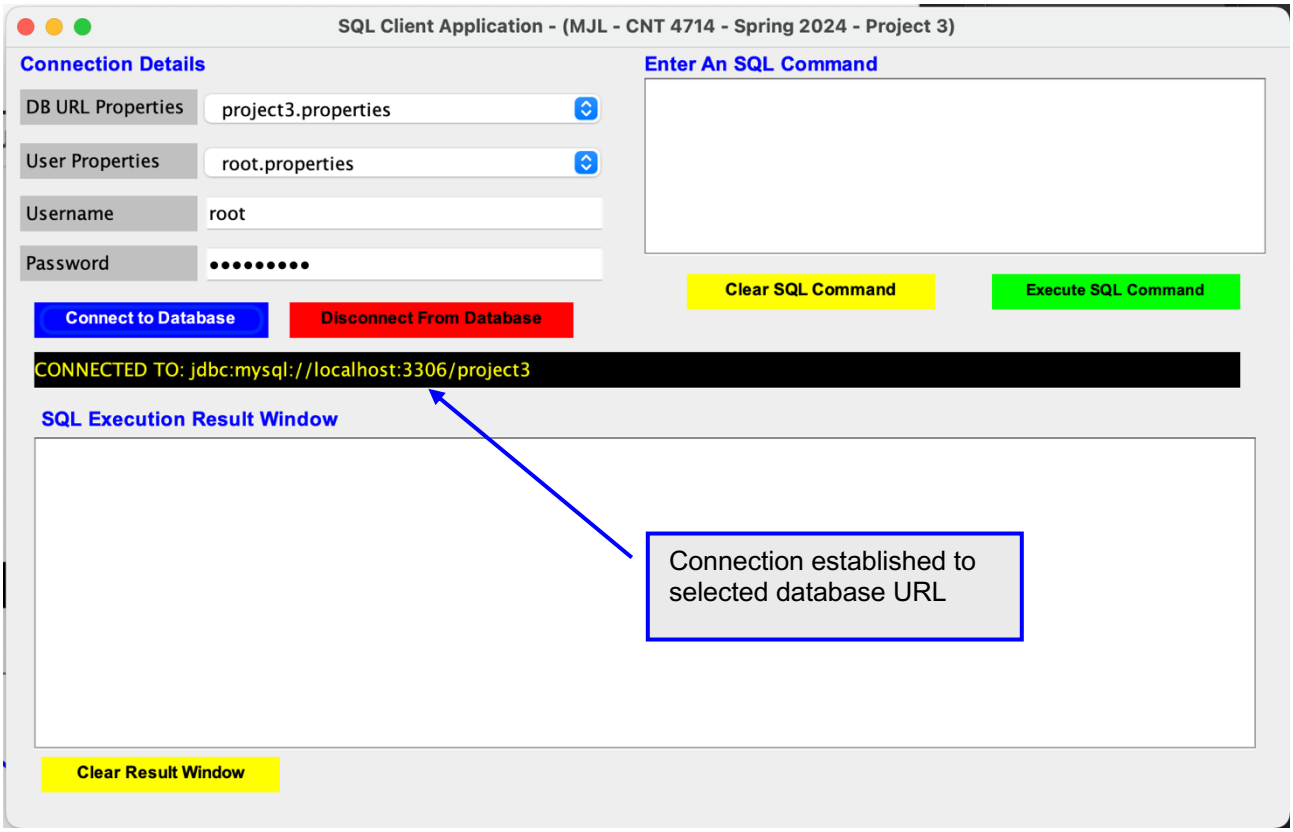
The remainder of the document illustrates the application at various phases during execution.

Many features and nuances of this project will be discussed in detail in the upcoming Q&A sessions. This document does **NOT** reflect every detail of this project, so pay attention to the Q&A session videos for the discussion of the various implementation details.

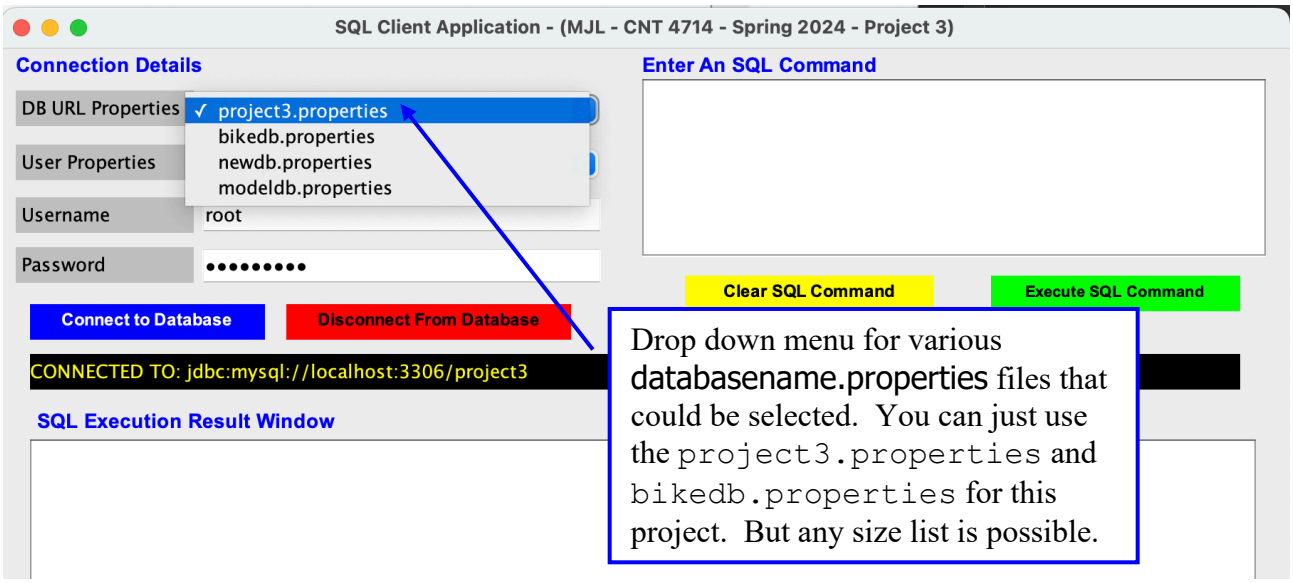
1. The GUI areas defined.



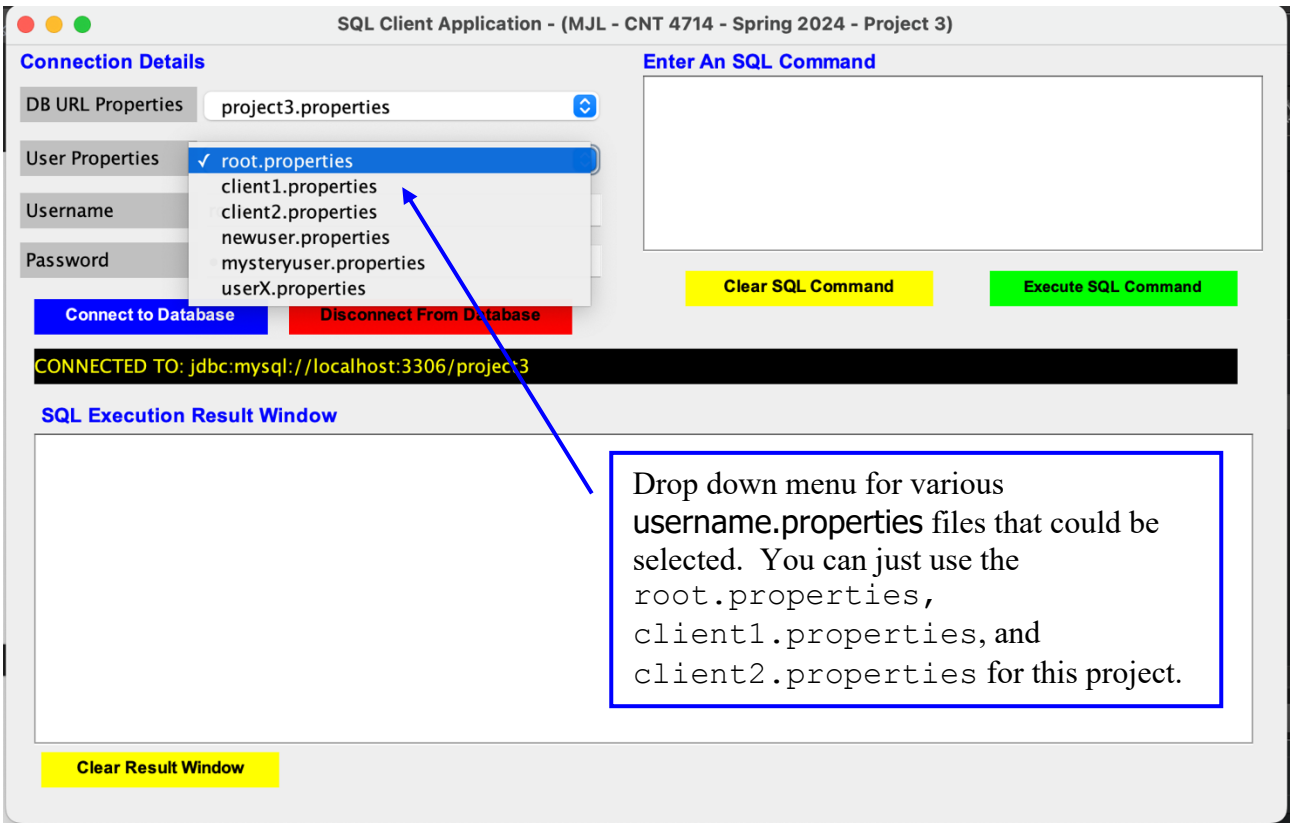
2. Screen shot illustrating an initial connection.



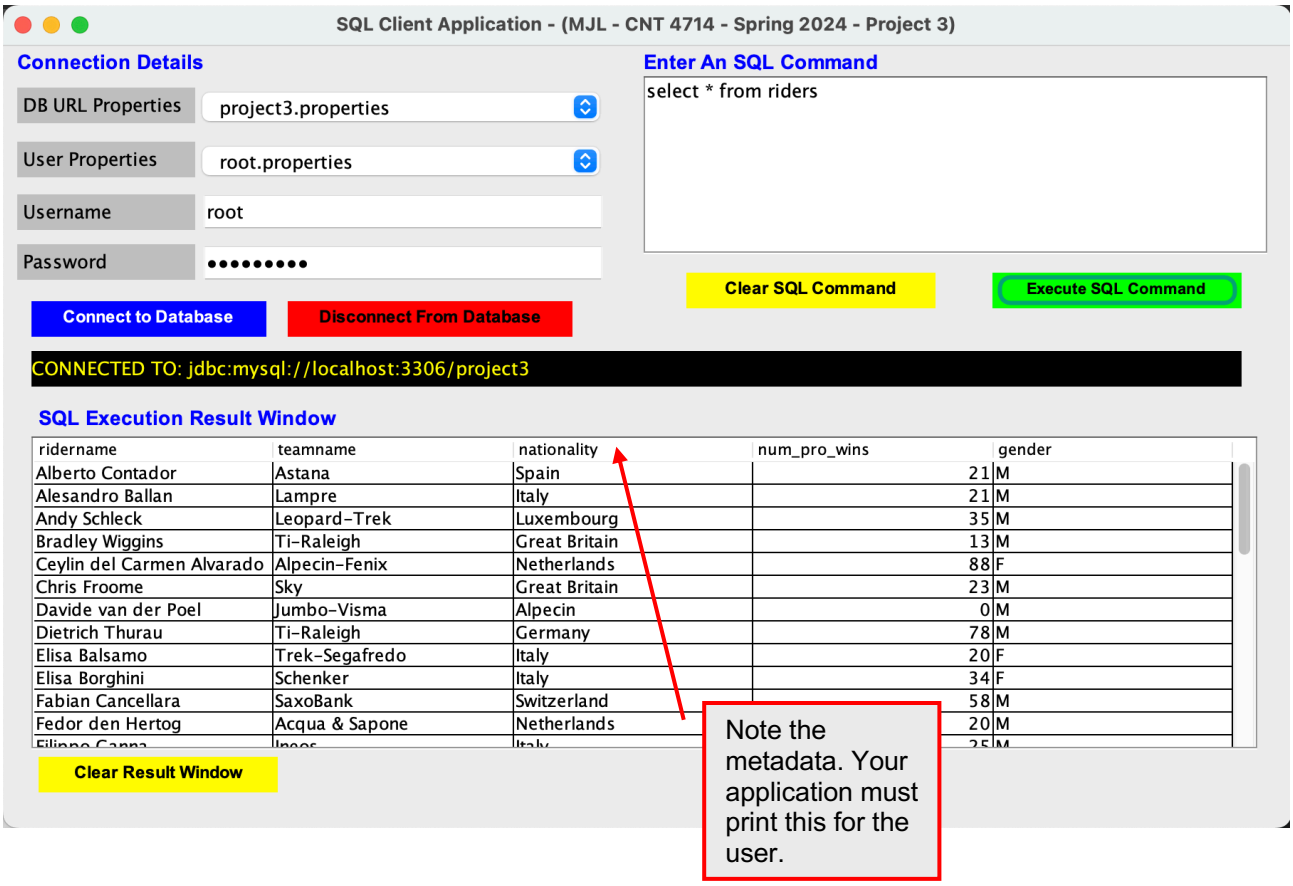
Illustrating the drop-down list of possible databasename.properties files that could be selected.



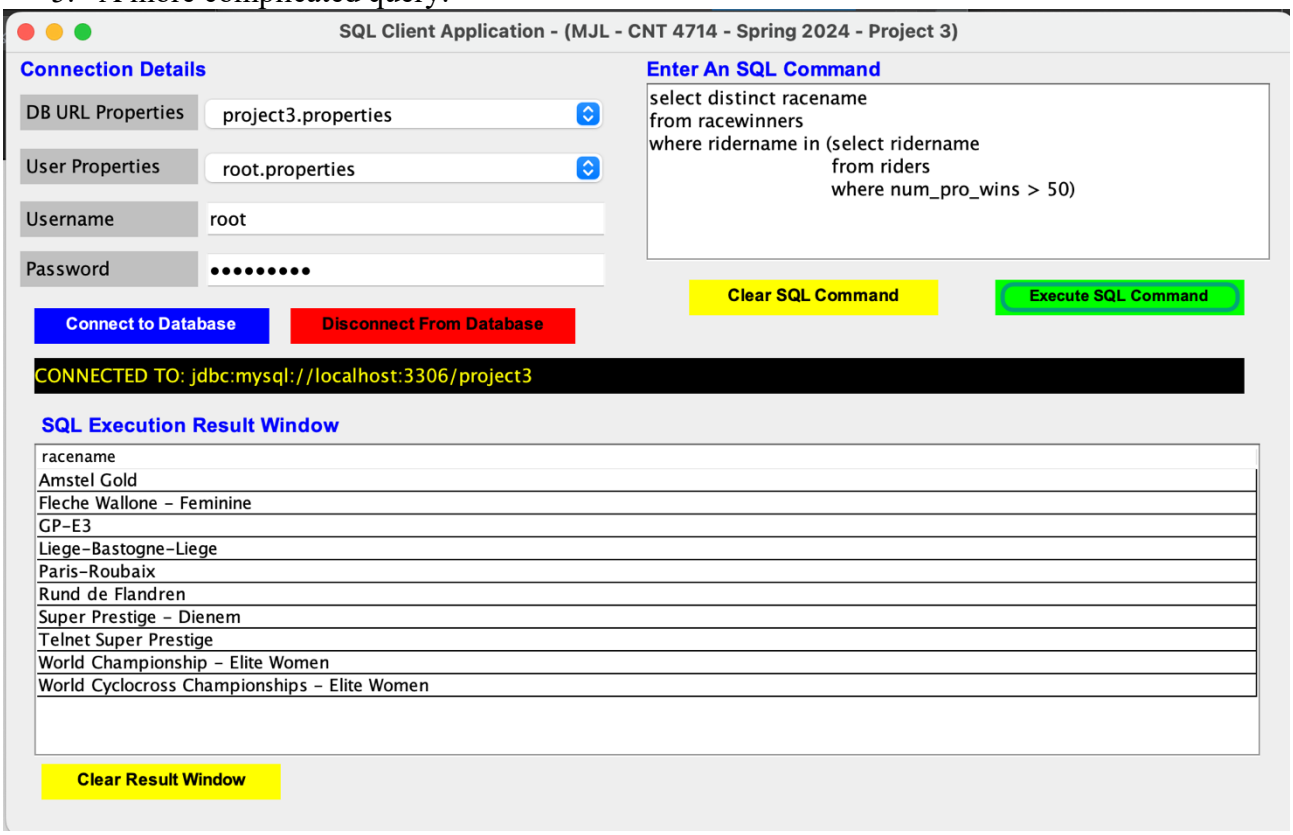
3. Illustrating the drop-down list of possible username.properties files that could be selected.



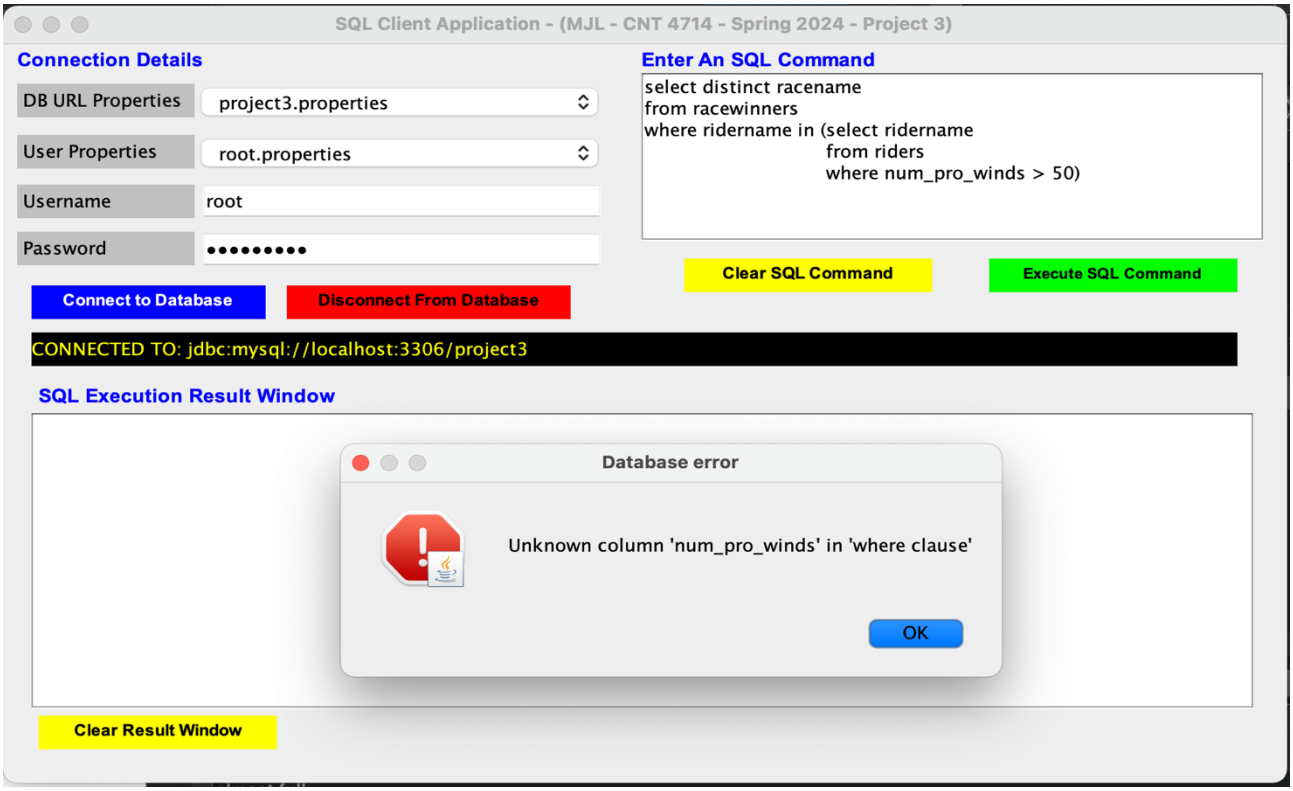
4. The root user has connected to the project3 database and issued a select command. Results are displayed in the SQL Execution Result Window.



5. A more complicated query:

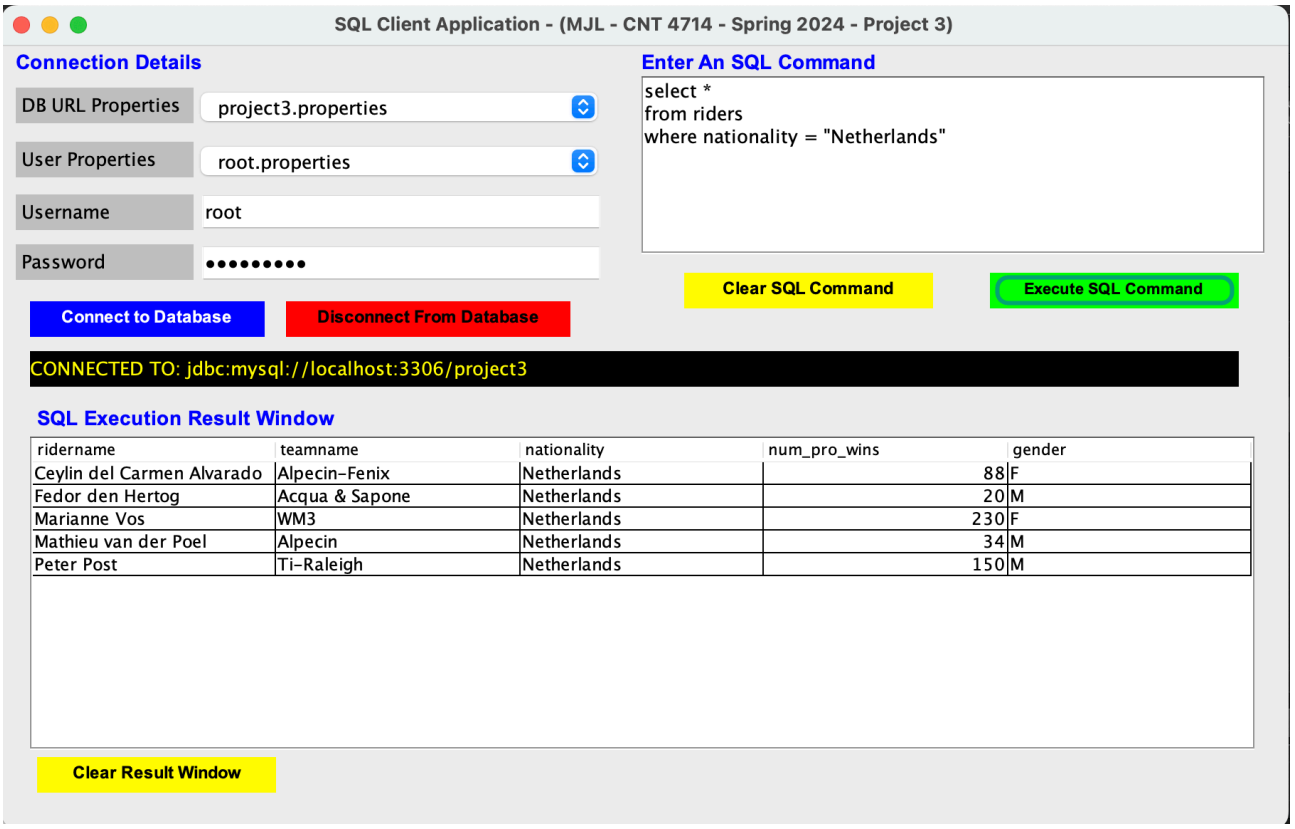


6. When the user makes a mistake entering a SQL command:

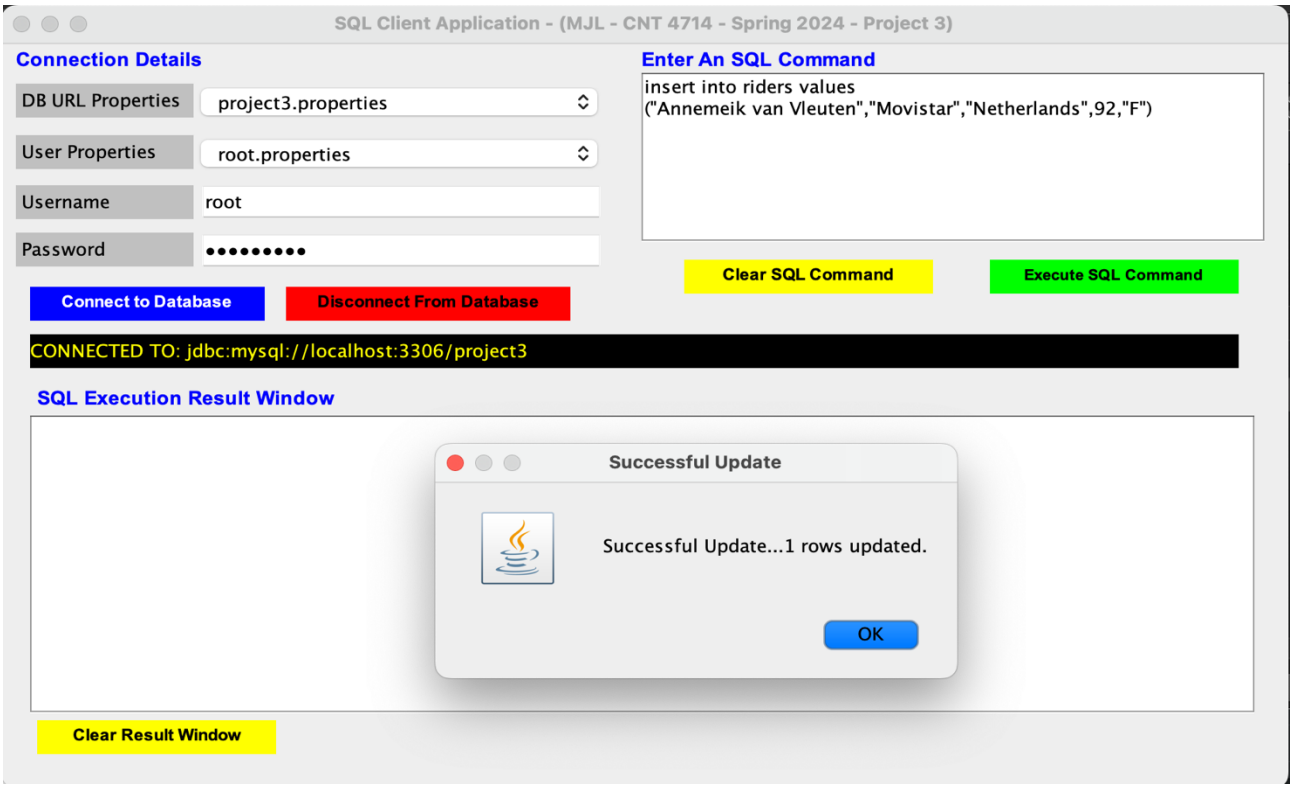


7. The following three screen shots illustrate that your application should be able to handle non-query commands from the users.

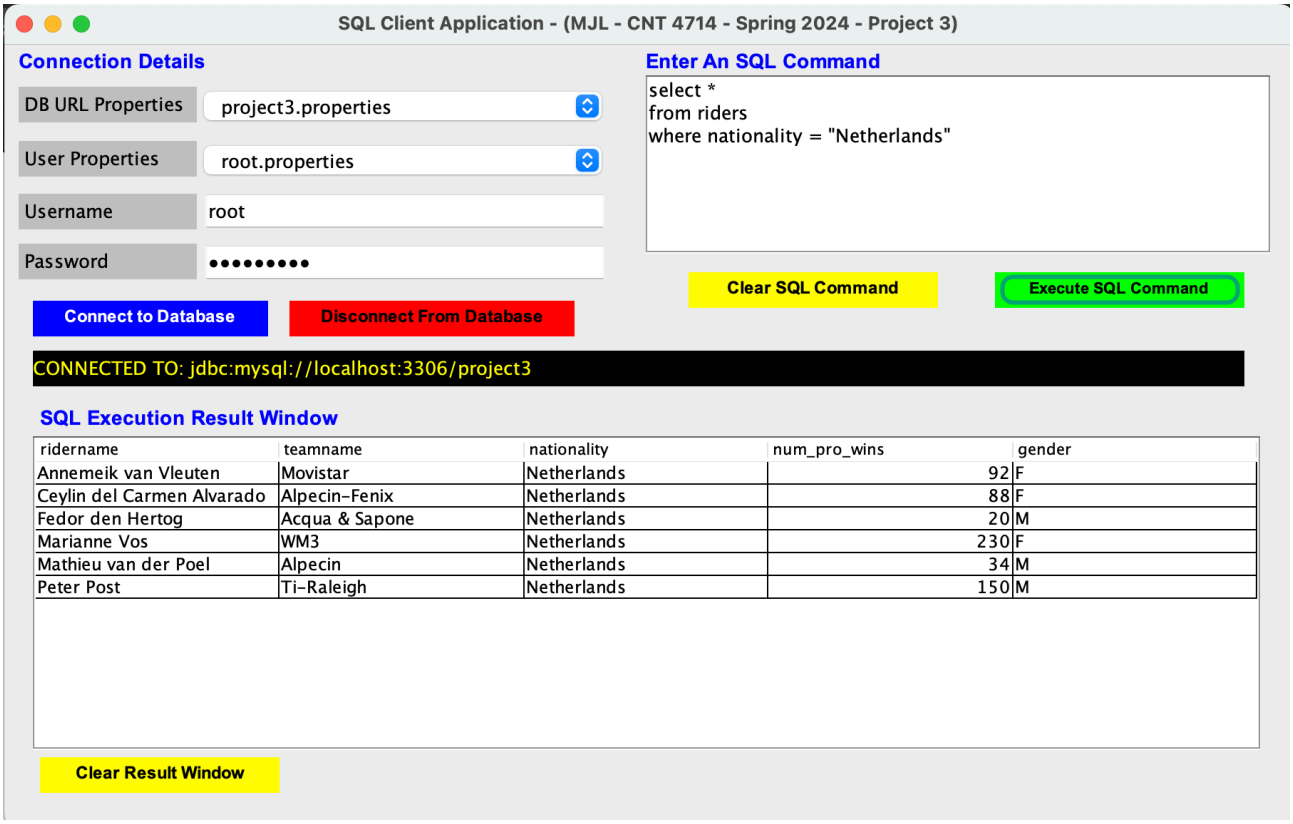
Before screen shot of a subset of the riders relation:



Insert command issued:

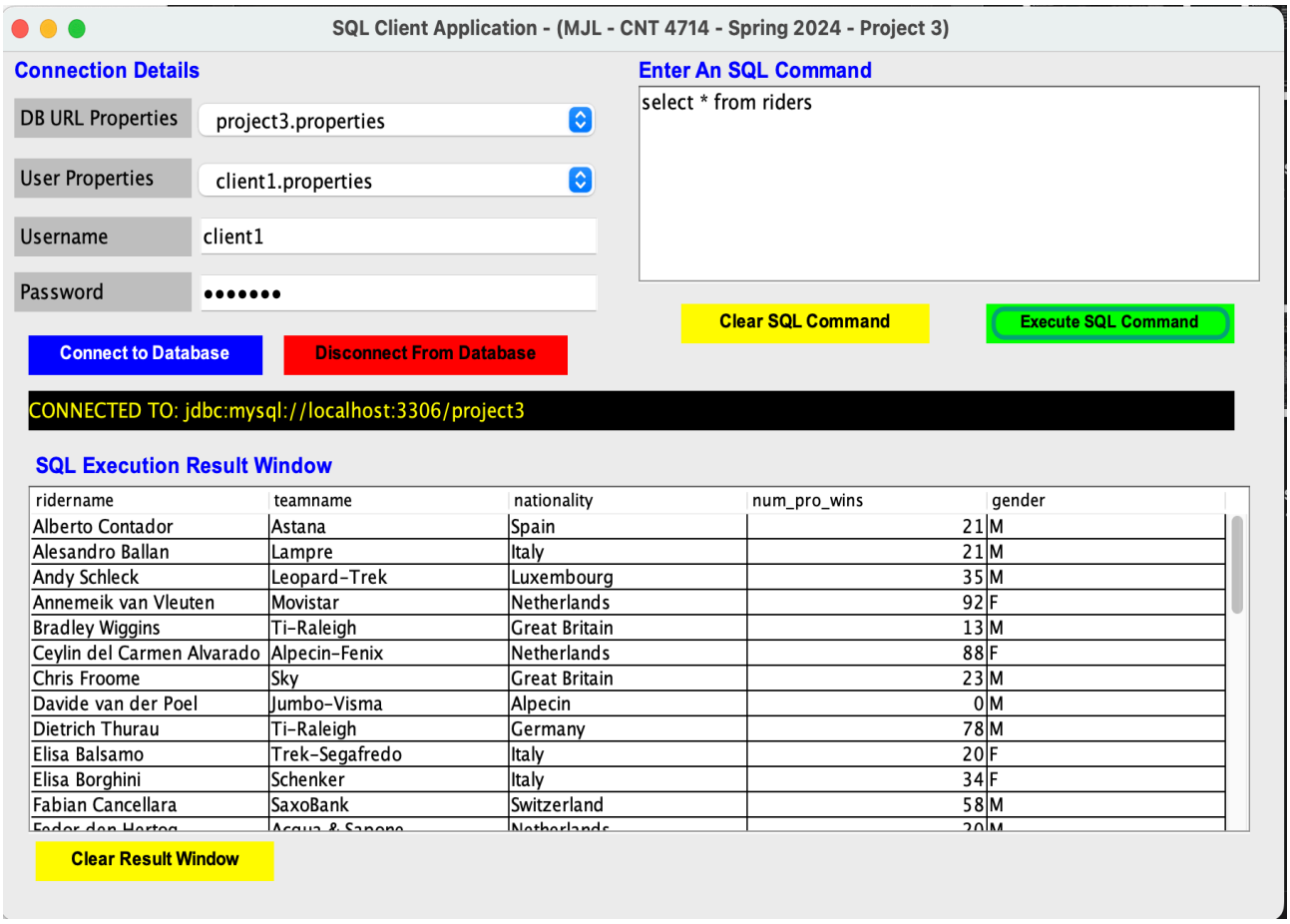


After screen shot of subset of riders relation after insert command was issued:



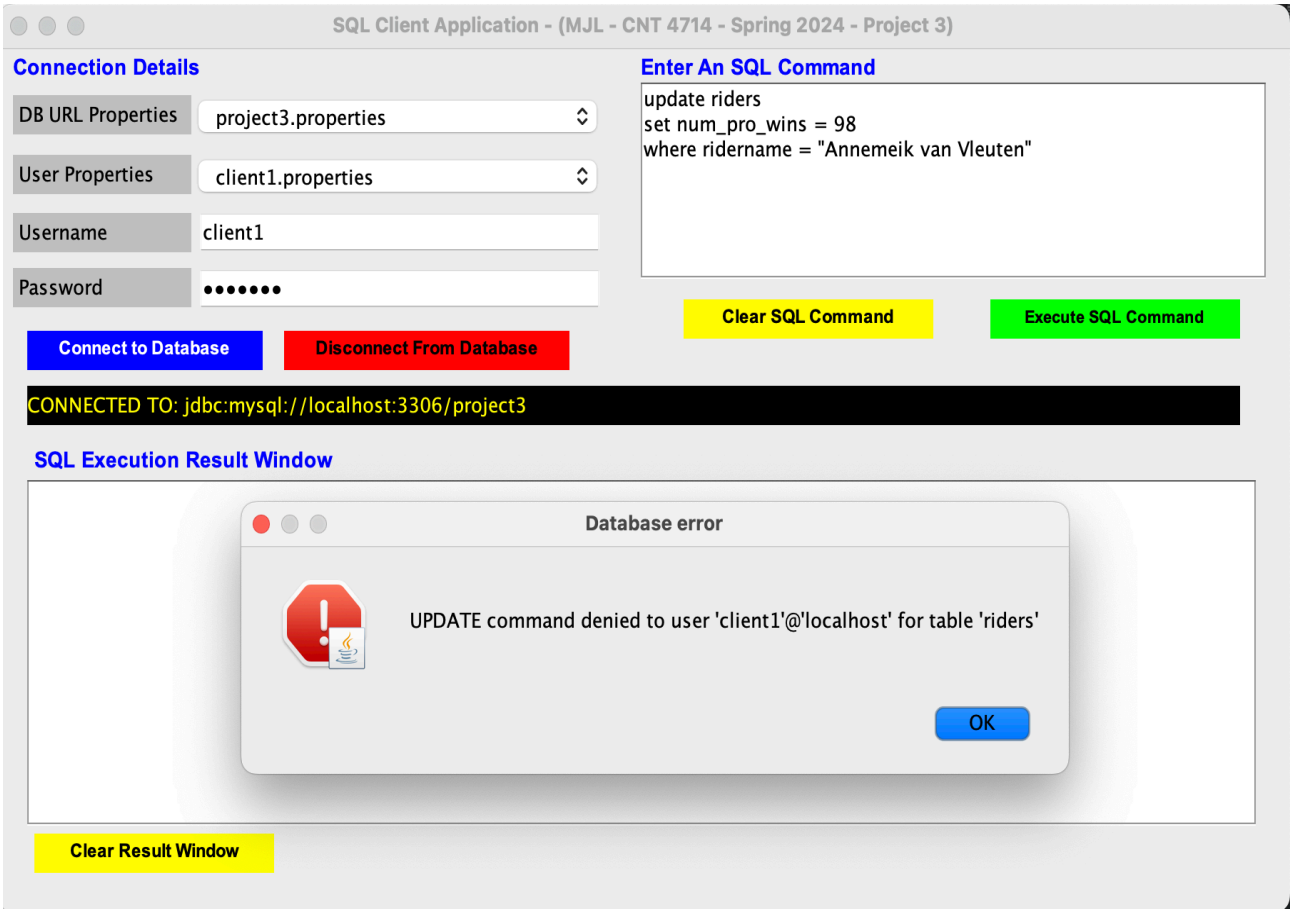
ridername	teamname	nationality	num_pro_wins	gender
Annemeik van Vleuten	Movistar	Netherlands	92	F
Ceylin del Carmen Alvarado	Alpecin-Fenix	Netherlands	88	F
Fedor den Hertog	Acqua & Sapone	Netherlands	20	M
Marianne Vos	WM3	Netherlands	230	F
Mathieu van der Poel	Alpecin	Netherlands	34	M
Peter Post	Ti-Raleigh	Netherlands	150	M

8. Screen shot illustrating the client1 user issuing a select command.

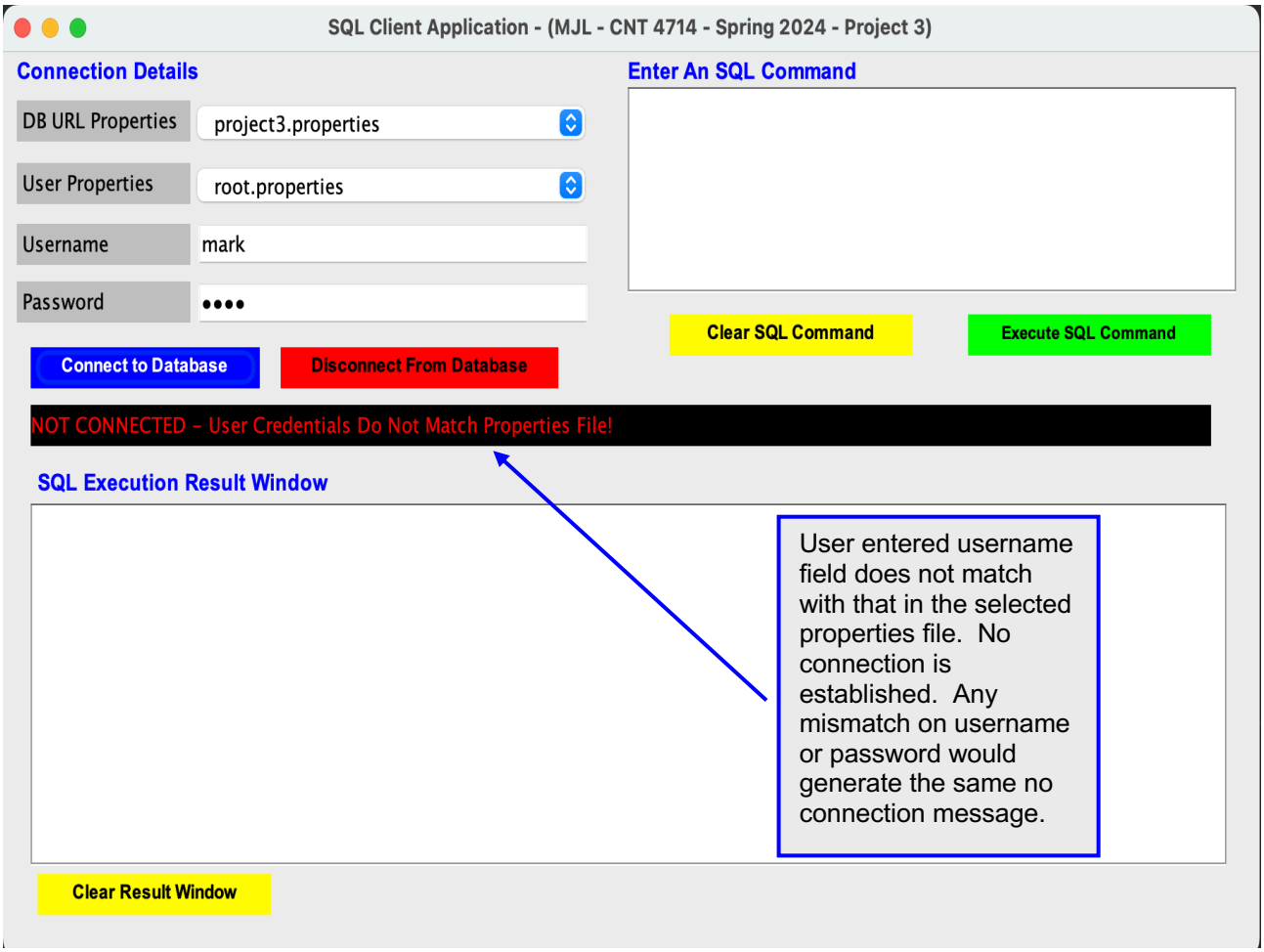


ridername	teamname	nationality	num_pro_wins	gender
Alberto Contador	Astana	Spain	21	M
Alesandro Ballan	Lampre	Italy	21	M
Andy Schleck	Leopard-Trek	Luxembourg	35	M
Annemeik van Vleuten	Movistar	Netherlands	92	F
Bradley Wiggins	Ti-Raleigh	Great Britain	13	M
Ceylin del Carmen Alvarado	Alpecin-Fenix	Netherlands	88	F
Chris Froome	Sky	Great Britain	23	M
Davide van der Poel	Jumbo-Visma	Alpecin	0	M
Dietrich Thurau	Ti-Raleigh	Germany	78	M
Elisa Balsamo	Trek-Segafredo	Italy	20	F
Elisa Borghini	Schenker	Italy	34	F
Fabian Cancellara	SaxoBank	Switzerland	58	M

9. Screen shot illustrating the client user issuing a command for which they do not have permission:

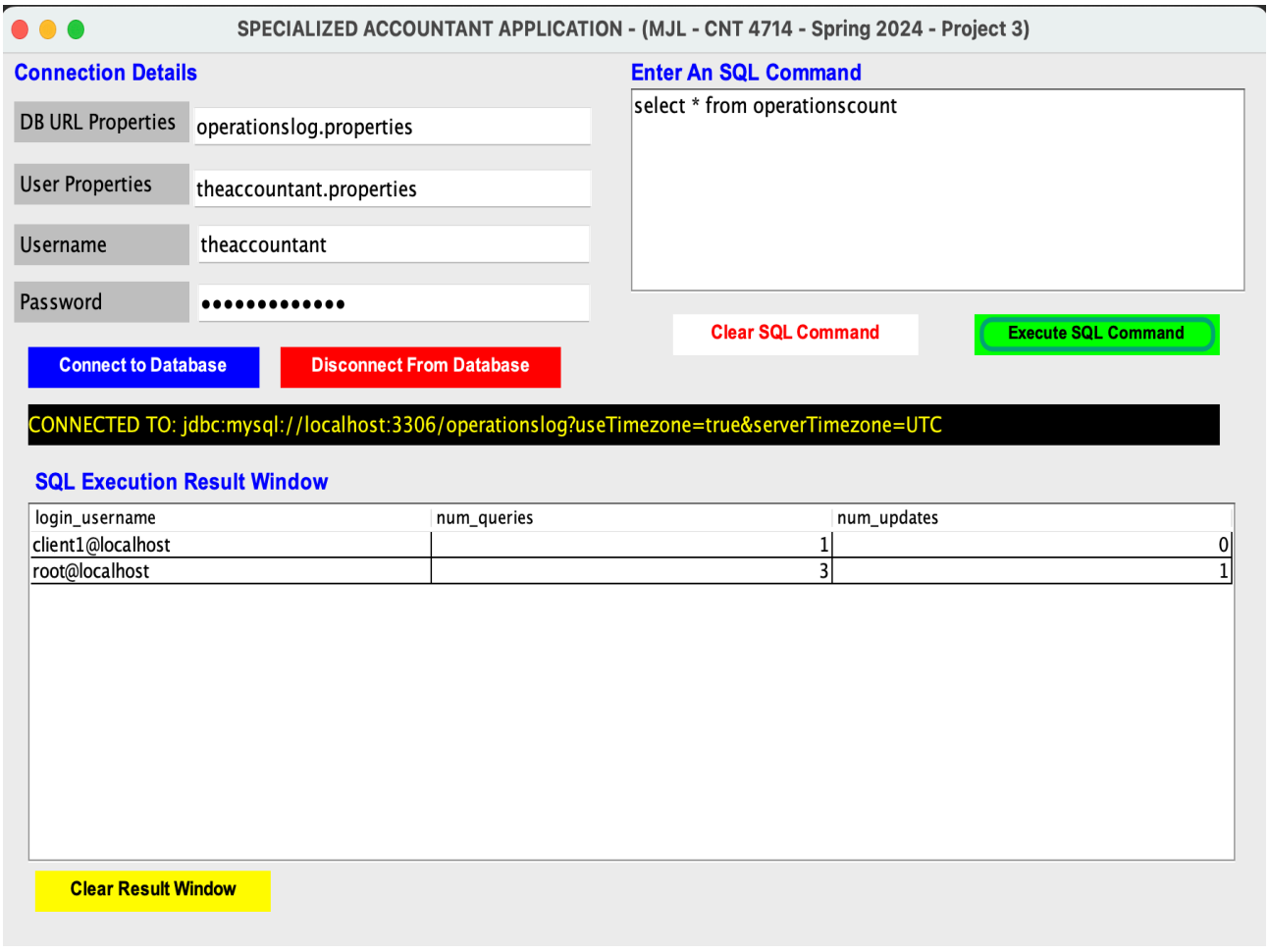


10. The following screenshot illustrates how the user-entered credentials must match those in the selected properties file in order to establish a connection to the database specified in the properties file.

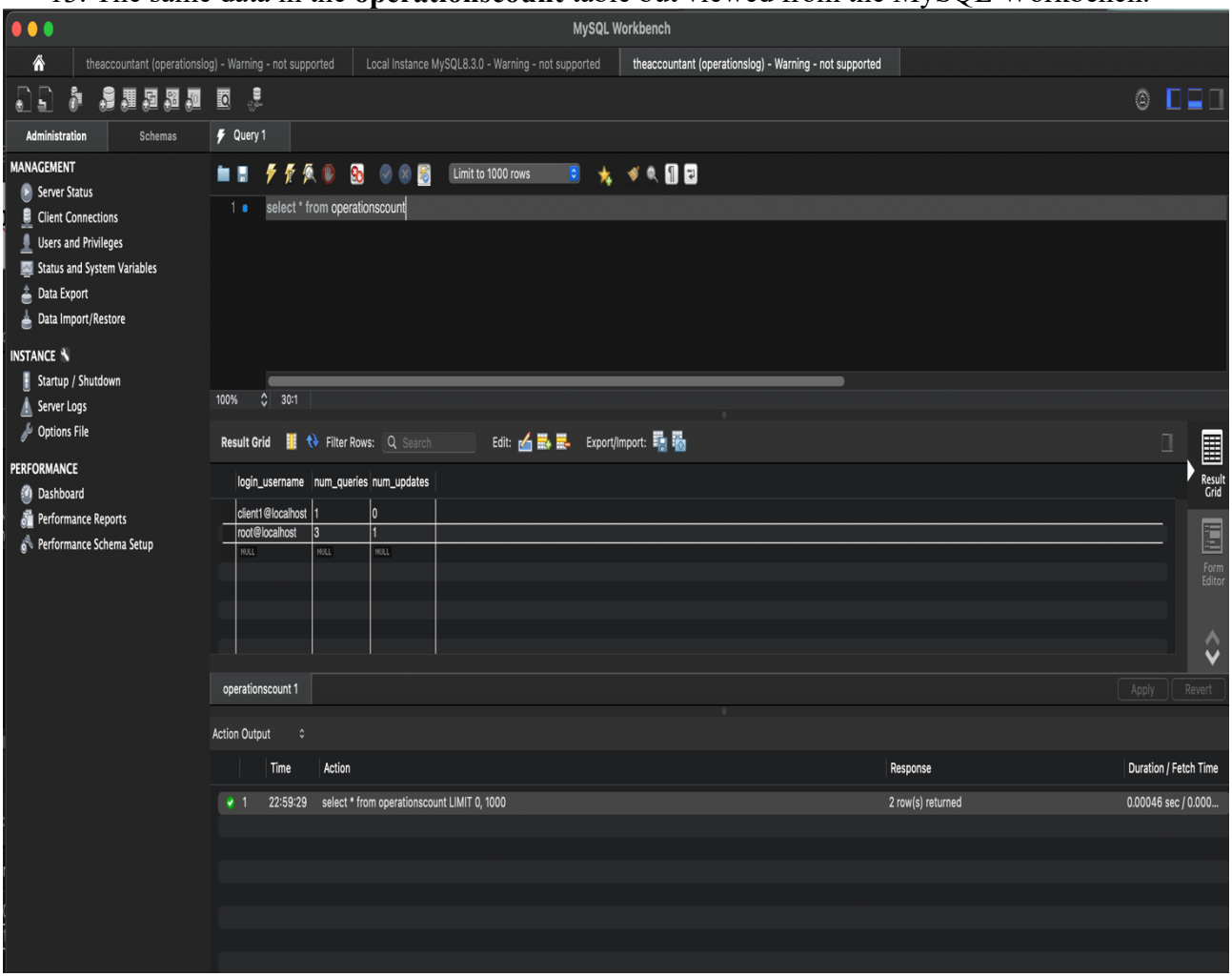


11. The specialized interface for theaccountant user is just a slightly modified version of the primary interface that has been illustrated in all of the previous images. This GUI does not allow the user to select either of the properties files that will be read. In this case, those values are hardcoded into the app and the user will only enter their login credentials for verification purposes. The same procedure will happen in that the user's credentials will be verified for a match with what is read from **theaccountant.properties** file. As before, if the credentials do not match, then no connection is established. Assuming the credentials match, then a connection to the **operationslog** database is established and theaccountant user can issue queries against this database. Only selection privilege is granted to theaccountant.

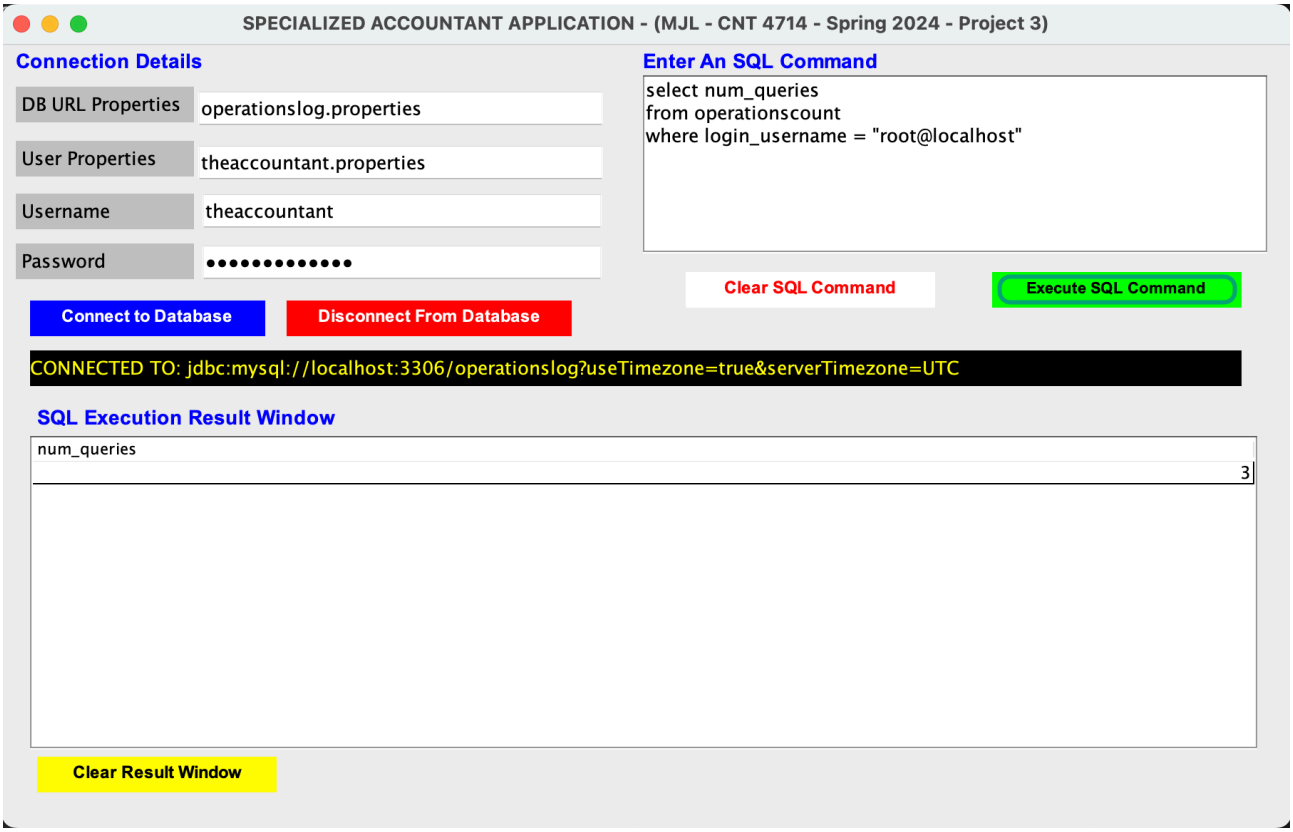
12. The following screenshot illustrates the **operationscount** table values after various operations have been completed. This screenshot is taken from the theaccountant user in the specialized interface created for the accountant user. We will discuss this interface in more detail in the Q&A sessions. You can see the similarity of this interface to that of your main application UI.



13. The same data in the **operationscount** table but viewed from the MySQL Workbench.



14. This example shows a different query issued by theaccountant user.



15. Below is a high-level time line for the project:

Step	Action
1	Make sure your MySQL server is correctly installed, configured, and operational before attempting any part of this project. Test root user connections by running some of the examples from the JDBC notes.
2	Create the three databases used in this project by executing their creation/population scripts from the Workbench.
3	Create the different user accounts on the MySQL server with the specific permissions granted to each different user account.
4	Make sure you understand completely how the DisplayQueryResults application and the helper class ResultSetTableModel work. These two classes can be used to develop your applications. Use and modify these classes as needed.
5	Start developing the main application. Construct the GUI and work on connection details first. Once connection issues are handled, then move on to executing commands on the DB server and getting results back and displayed. Use the root user for development purposes since they have all permissions and any correct command will execute.
6	Once main application is done, build the specialize app for theaccountant user.
7	Modify the main application to handle logging of user commands.
8	Finish early – you have a lot of screenshots to take and label.