

CNT 4714 – Project Two – Summer 2021

Title: “Project Two: Two-Tier Client-Server Application Development With MySQL and JDBC”

Points: 100 points

Due Date: Sunday June 27, 2021 by 11:59 pm (WebCourses Time)

Objectives: To develop a two-tier Java based client-server application 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 assignment you will develop a Java-based GUI front-end (client-side) application that will connect to your MySQL server via JDBC.

You are to develop a Java application that will allow any client (the end-user) to execute commands against the database. You will create a Java GUI-based application 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 client. 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. Note also, that if the user wishes to change databases in the middle of a session, they must reconnect to the new database. Their user information can remain in the proper window, but you must click the reconnect button to establish a connection to the new database. You will be able to start multiple instances of your Java application and allow different clients to connect simultaneously to the MySQL server, since the default number of connections is set at 151 (see your Workbench options file under the networking tab). In addition, a transaction logging operation will occur which keeps a running total of the number of queries and the number of updates that have occurred via the user application. This is a separate database (i.e., completely different database than any that the user can connect to), that the application will connect, with root user privileges, and update after each user operation completes. See below for more details on this feature of your application.

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

References for this assignment:

Notes: Lecture Notes for MySQL and JDBC.

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 assignment page and is named “`project2dbscript.sql`”. This script creates a database named **project2**. You can use the MySQL Workbench for this step, or the command line whichever you prefer.

The **second step** is to create authorizations for a client user (in addition to the root user) named `project2client`. By default your root user has all permissions on the **project2** 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 check and set permissions for the client as follows:

Register the new user named **project2client** and assign them the password *P2client@*, and assign to this user only selection privileges on the **project2** schema.

The **third step** is to create the **operationslog** database using the `project2operationslog.sql` script. This script file is also available on WebCourses.

Output Specification: There are three parts for the output for this project. 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: **project2rootuserscript.sql** available on the course website. There are eight different commands in this script and some of the commands will have more than one output capture (see below). 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: **project2clientuserscript.sql** available on the course website. There are three different commands in this script and some of the commands will have more than one output capture (see below). Part 3 is to provide a screenshot of the **operationscount** table, taken from the output of the execution of the query: `select * from operationscount;`, from the MySQL Workbench app (see page 10 for an example of this screenshot).

To produce your final output, first recreate the database, then run the root user commands followed by the client commands.

Deliverables:

Zip up all of the .java files associated with your application as well as the screen shots from each of the commands specified in both the **project2rootuserscript.sql** and **project2clientuserscript.sql** files, and the screenshot of the final status of the **operationscount** table, and submit them via WebCourses no later than 11:59pm Sunday June 27, 2021. Be sure to clearly label each screen shot.

Details:

Shown below is a screen shot of the initial GUI. Notice that there are drop-down lists for selecting the JDBC driver and database URL that the user must select. The user must also specify a username and password (MySQL option) before connecting to the database.

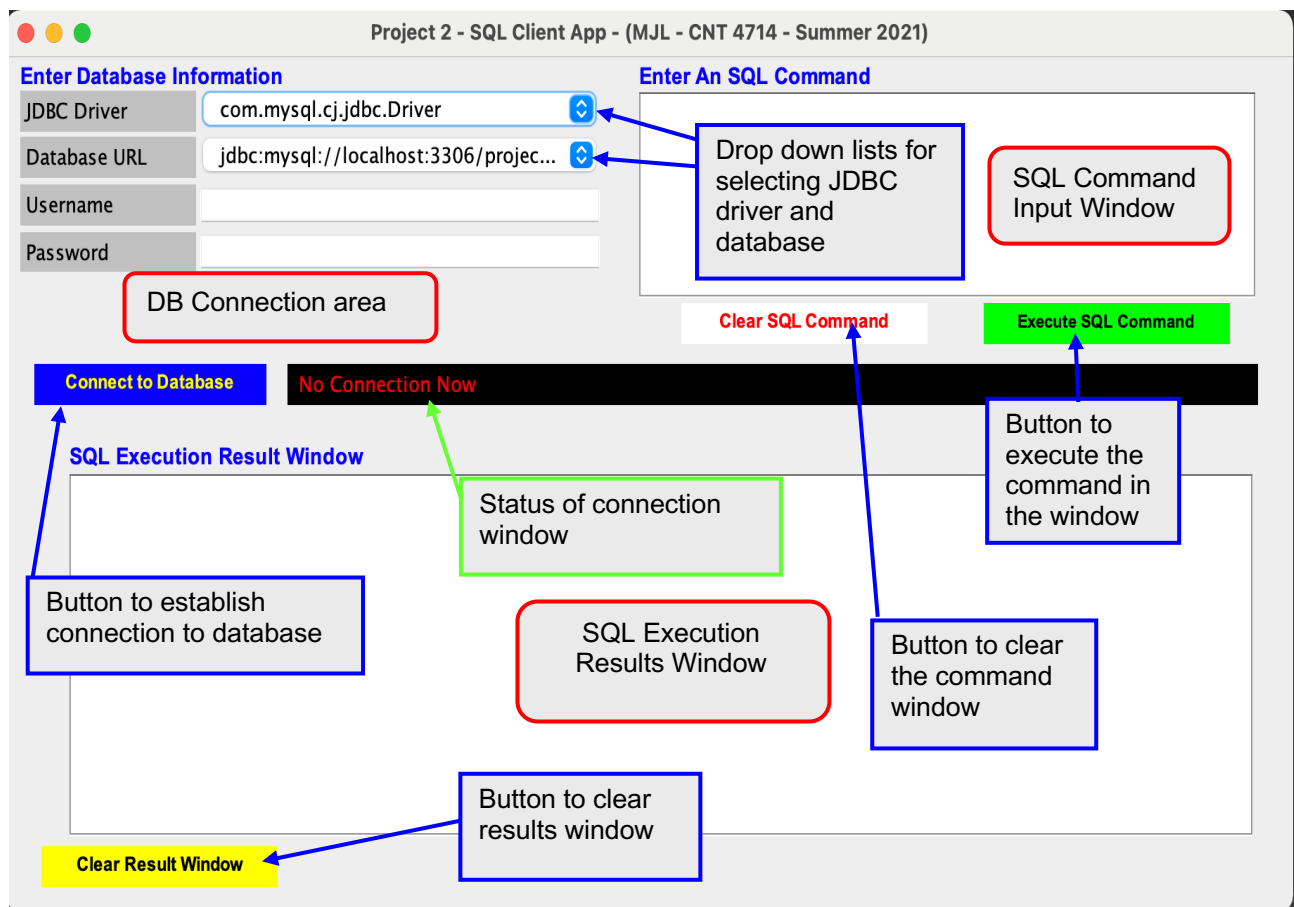
You should provide buttons for the user to clear the command window as well as the result window. The status of the connection should be returned to the GUI and displayed in the connection area.

The output of all SQL commands should be returned to the SQL Execution Result window. Please note that only SQL commands can be executed via this application, we will 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.)

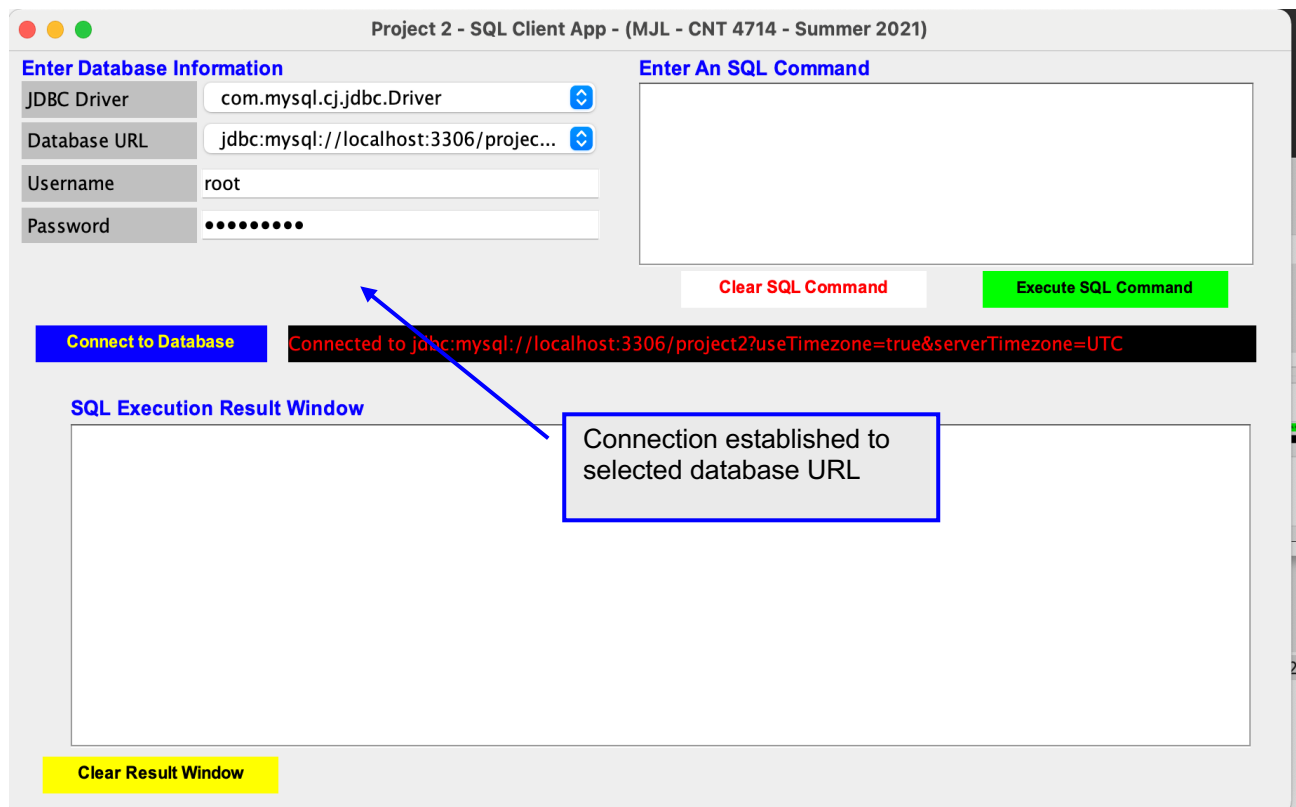
As each user command is executed (only successful commands) the **operationscount** table in the **operationslog** database must be updated by your application. Each query and each update will be logged (counted) separately. Only successfully operations will be logged – any transaction erroring will not increment any counter. Your application must obtain a connection to the **operationslog** database and perform the update as a root user. These operations are invisible to the end user running the application. The app should connect to the **operationslog** database using a properties file.

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 7-8 for an illustration of this.

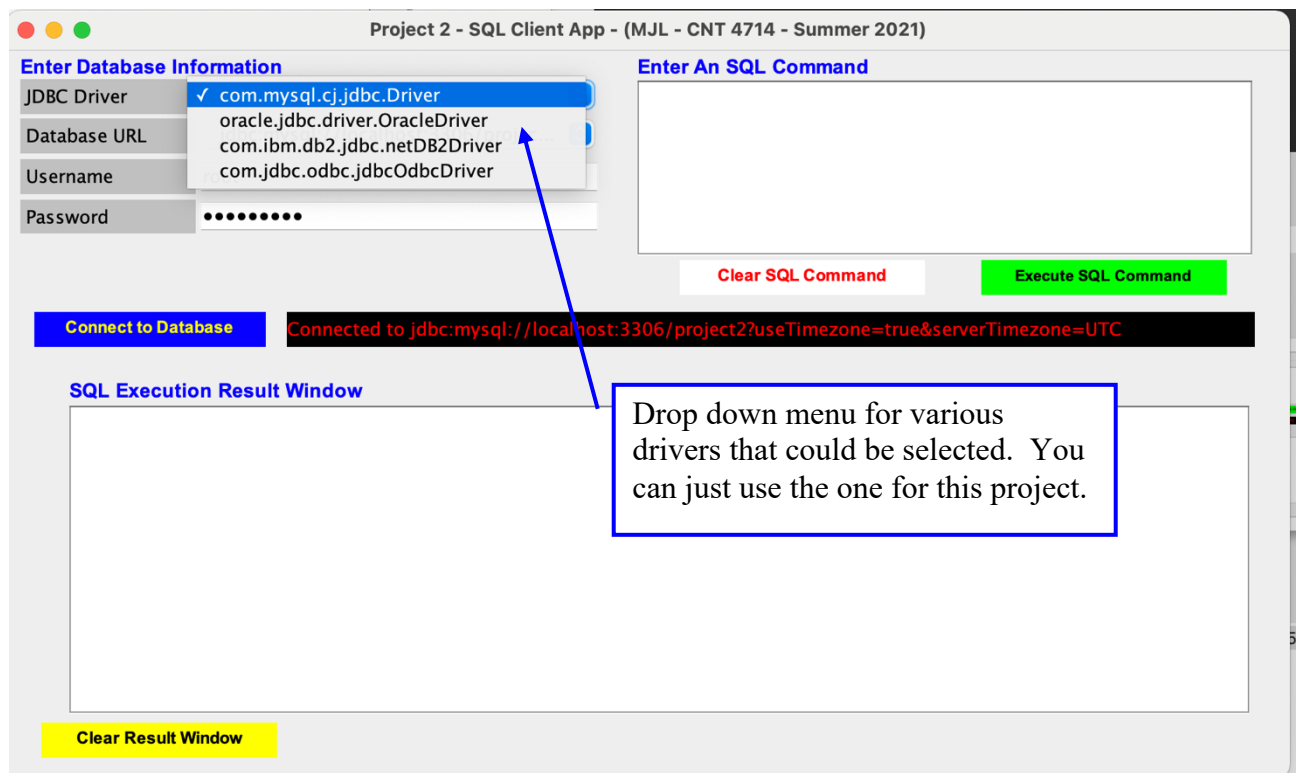
The GUI areas defined.



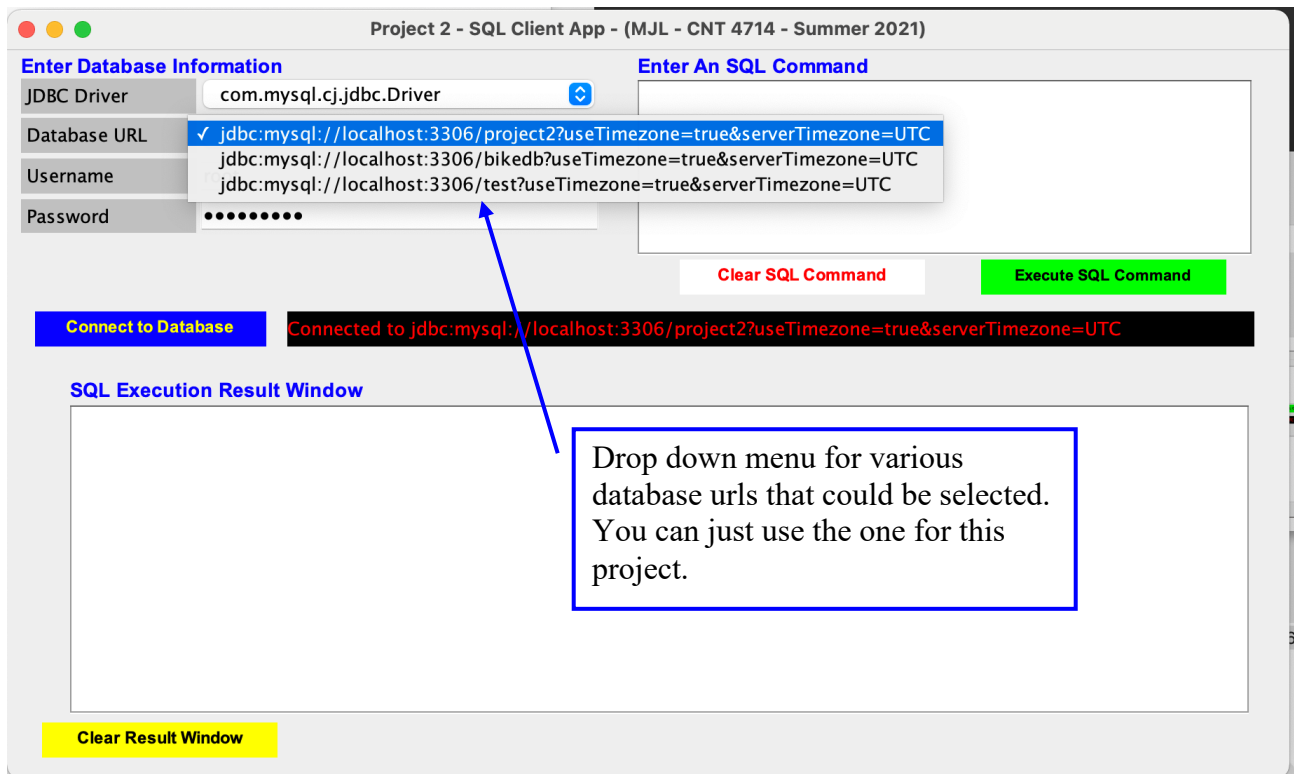
Screen shot illustrating an initial connection.



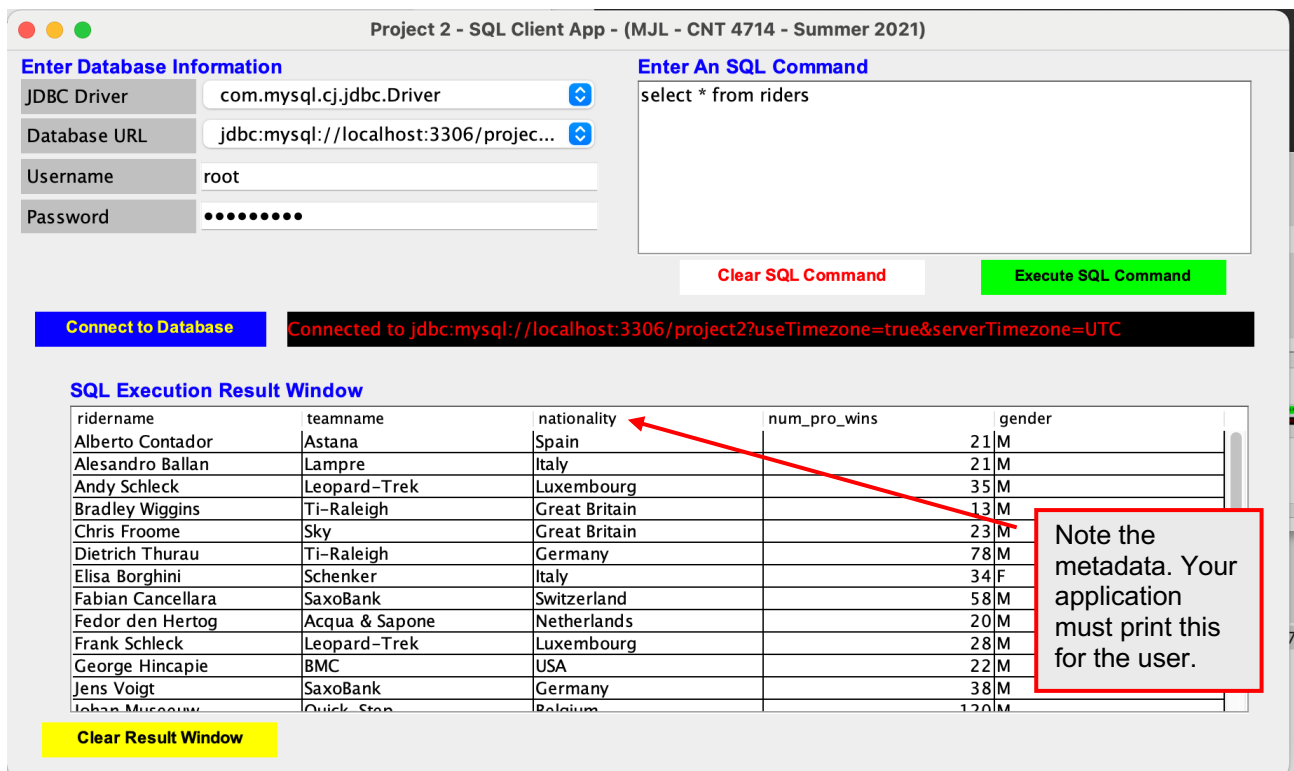
Illustrating the drop-down list of possible drivers that could be selected.



Illustrating the drop-down list of possible database URLs available.



User has connected to a database and issued a select command. Results are displayed in the SQL Execution Window.



A more complicated query:

Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)

Enter Database Information

JDBC Driver: com.mysql.cj.jdbc.Driver
Database URL: jdbc:mysql://localhost:3306/projec...
Username: root
Password:

Enter An SQL Command

```
select distinct racename
from racewinners
where ridername in (select ridername
                    from riders
                    where num_pro_wins > 50)
```

Clear SQL Command Execute SQL Command

Connect to Database Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC

SQL Execution Result Window

racename
Amstel Gold
Fleche Wallone - Feminine
GP-E3
Liege-Bastogne-Liege
Paris-Roubaix
Rund de Flandren
World Championship - Elite Women

Clear Result Window

When the user makes a mistake entering a SQL command:

Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)

Enter Database Information

JDBC Driver: com.mysql.cj.jdbc.Driver
Database URL: jdbc:mysql://localhost:3306/projec...
Username: root
Password:

Enter An SQL Command

```
select distinct racename
from racewinners
where ridername in (select ridername
                    from riders
                    where num_pro_winds > 50)
```

Clear SQL Command Execute SQL Command

Connect to Database Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC

SQL Execution Result Window

Database error

Unknown column 'num_pro_winds' in 'where clause'

OK

Clear Result Window

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:

The screenshot shows a window titled "Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)". The interface is divided into several sections:

- Enter Database Information:** Contains four input fields: "JDBC Driver" (com.mysql.cj.jdbc.Driver), "Database URL" (jdbc:mysql://localhost:3306/projec...), "Username" (root), and "Password" (masked with dots).
- Enter An SQL Command:** A text area containing the SQL query:

```
select *  
from riders  
where nationality = "Holland"
```
- Buttons:** Below the SQL command area are two buttons: "Clear SQL Command" (red) and "Execute SQL Command" (green).
- Connect to Database:** A blue button on the left, and a black status bar on the right showing "Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC".
- SQL Execution Result Window:** A table displaying the results of the query. The table has five columns: ridername, teamname, nationality, num_pro_wins, and gender. The first row shows data for Marianne Vos.
- Clear Result Window:** A yellow button at the bottom left of the result window.

ridername	teamname	nationality	num_pro_wins	gender
Marianne Vos	WM3	Holland	230	F

Insert command issued:

Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)

Enter Database Information

JDBC Driver: com.mysql.cj.jdbc.Driver
Database URL: jdbc:mysql://localhost:3306/projec...
Username: root
Password:

Enter An SQL Command

insert into riders values
("Annemeik van Vlueten", "Moviestar", "Holland", 88, "F")

Clear SQL Command Execute SQL Command

Connect to Database Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC

SQL Execution Result Window

Clear Result Window

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

Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)

Enter Database Information

JDBC Driver: com.mysql.cj.jdbc.Driver
Database URL: jdbc:mysql://localhost:3306/projec...
Username: root
Password:

Enter An SQL Command

select *
from riders
where nationality = "Holland"

Clear SQL Command Execute SQL Command

Connect to Database Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC

SQL Execution Result Window

ridername	teamname	nationality	num_pro_wins	gender
Annemeik van Vlueten	Moviestar	Holland	88	F
Marianne Vos	WM3	Holland	230	F

Clear Result Window

Screen shot illustrating the client user issuing a select command. Note new connection.

Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)

Enter Database Information

JDBC Driver:

Database URL:

Username:

Password:

Enter An SQL Command

Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC

SQL Execution Result Window

ridername	teamname	nationality	num_pro_wins	gender
Alberto Contador	Astana	Spain	21	M
Alessandro Ballan	Lampre	Italy	21	M
Andy Schleck	Leopard-Trek	Luxembourg	35	M
Annemeik van Vlueten	Movistar	Holland	88	F
Bradley Wiggins	Ti-Raleigh	Great Britain	13	M
Chris Froome	Sky	Great Britain	23	M
Dietrich Thurau	Ti-Raleigh	Germany	78	M
Elisa Borghini	Schenger	Italy	34	F
Fabian Cancellara	SaxoBank	Switzerland	58	M
Fedor den Hertog	Acqua & Sapone	Netherlands	20	M
Frank Schleck	Leopard-Trek	Luxembourg	28	M
George Hincapie	BMC	USA	22	M
Jens Meier	SaxoBank	Germany	28	M

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

Project 2 - SQL Client App - (MJL - CNT 4714 - Summer 2021)

Enter Database Information

JDBC Driver:

Database URL:

Username:


Password:

Enter An SQL Command

Connected to jdbc:mysql://localhost:3306/project2?useTimezone=true&serverTimezone=UTC

SQL Execution Result Window

Database error

 UPDATE command denied to user 'project2client'@'localhost' for table 'riders'

Screen shot illustrating the **operationscount** table values after various operations have completed. Taken from a root user connection in the MySQL WorkBench using the **operationslog** database. Note that these are not the numbers you will expect to see after executing the root and client user scripts. This is just an example.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the Schemas tree with the following structure:

- bikedb
 - operationslog
 - Tables
 - operationscount
 - Views
 - Stored Procedures
 - Functions
- project2
 - Tables
 - bikes
 - racewinners
 - riders
 - teams
 - Views
 - Stored Procedures
 - Functions

The main editor shows a SQL query: `select * from operationscount;`. The Result Grid displays the following data:

num_queries	num_updates
16	3
NULL	NULL

The bottom panel, labeled "Action Output", shows a log of operations:

	Time	Action	Response	Duration / Fetch Time
✓	131 19:26:10	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00032 sec / 0.0000...
✓	132 19:26:37	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00036 sec / 0.0000...
✓	133 19:29:29	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00032 sec / 0.0000...
✓	134 19:32:21	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00036 sec / 0.0000...
✓	135 19:33:35	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00034 sec / 0.0000...
✓	136 19:36:12	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00037 sec / 0.0000...
✓	137 19:49:25	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00035 sec / 0.0000...
✓	138 20:05:46	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00032 sec / 0.0000...
✓	139 13:10:22	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00032 sec / 0.0000...
✓	140 13:45:12	select * from operationscount LIMIT 0, 1000	1 row(s) returned	0.00034 sec / 0.0000...

The status bar at the bottom indicates "Query Completed".