**Ben Brandhorst**

**November 30th 2019**

**Database Security**

**SDEV 350 7980**

**Professor Ivan De Los Santos**

**LAB 4 – Database Design for Secure Software Applications**

1. Demonstrate the SDEV350App.jar fails upon first test. Using your assigned Student account, sign in using your username and password. You should be able to connect (and disconnect) to the database without issue, but when you attempt to insert a record into the Students table or query the Students table you should receive an error message and not be successful. Begin documenting and fully describing in a word document the results of each of the following tests:

a. Launching the SDEV350App.jar

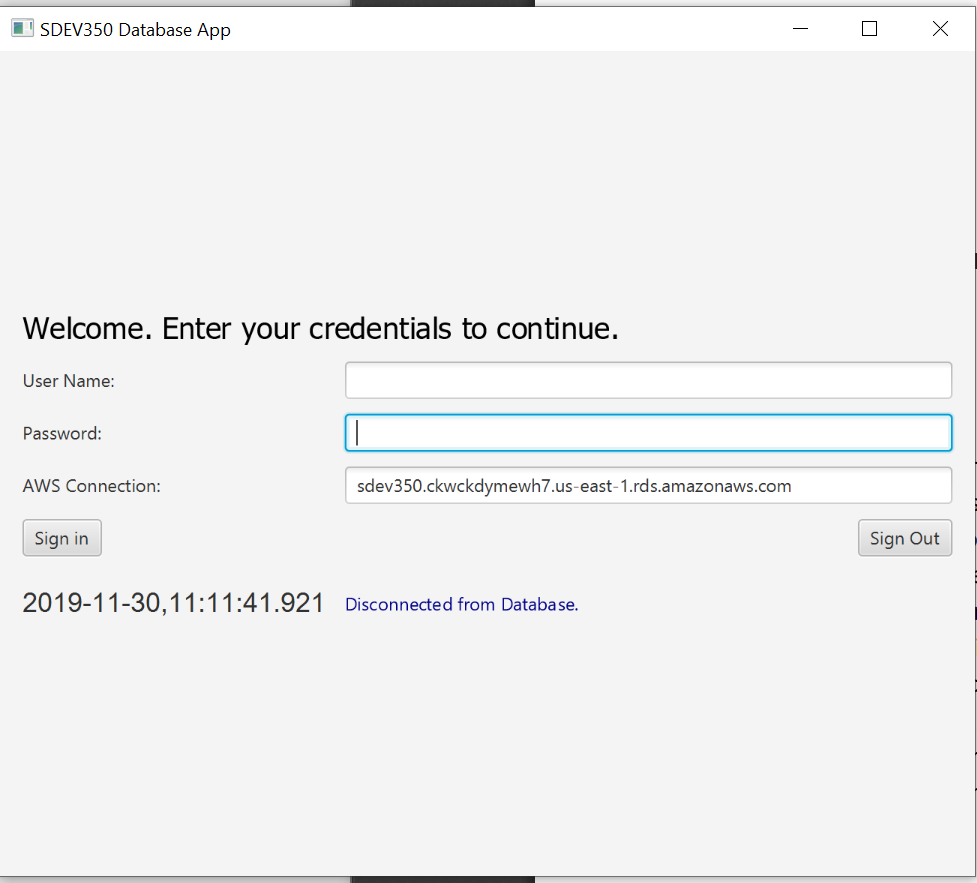
b. Connecting to the Oracle database by selecting “Sign in”

c. Entering a new Student record and selecting “Add Student”

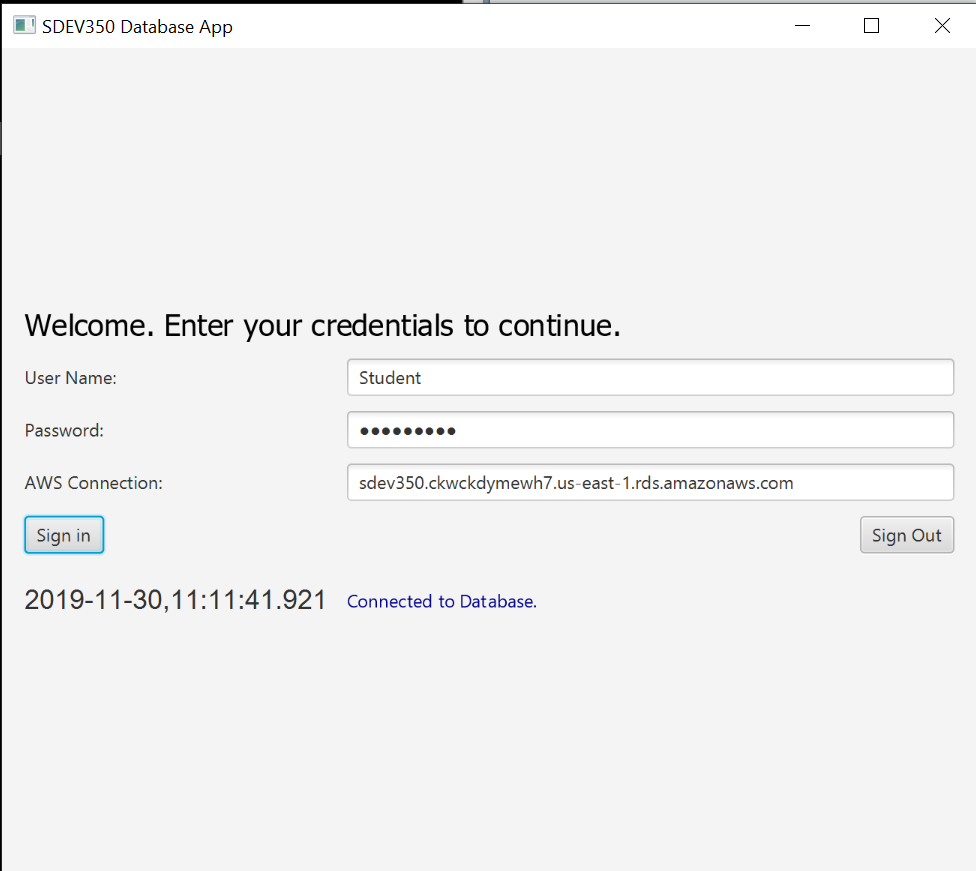
d. Querying the Students table by selecting “Query Students”

e. Disconnecting from the Oracle database by selecting “Sign out”

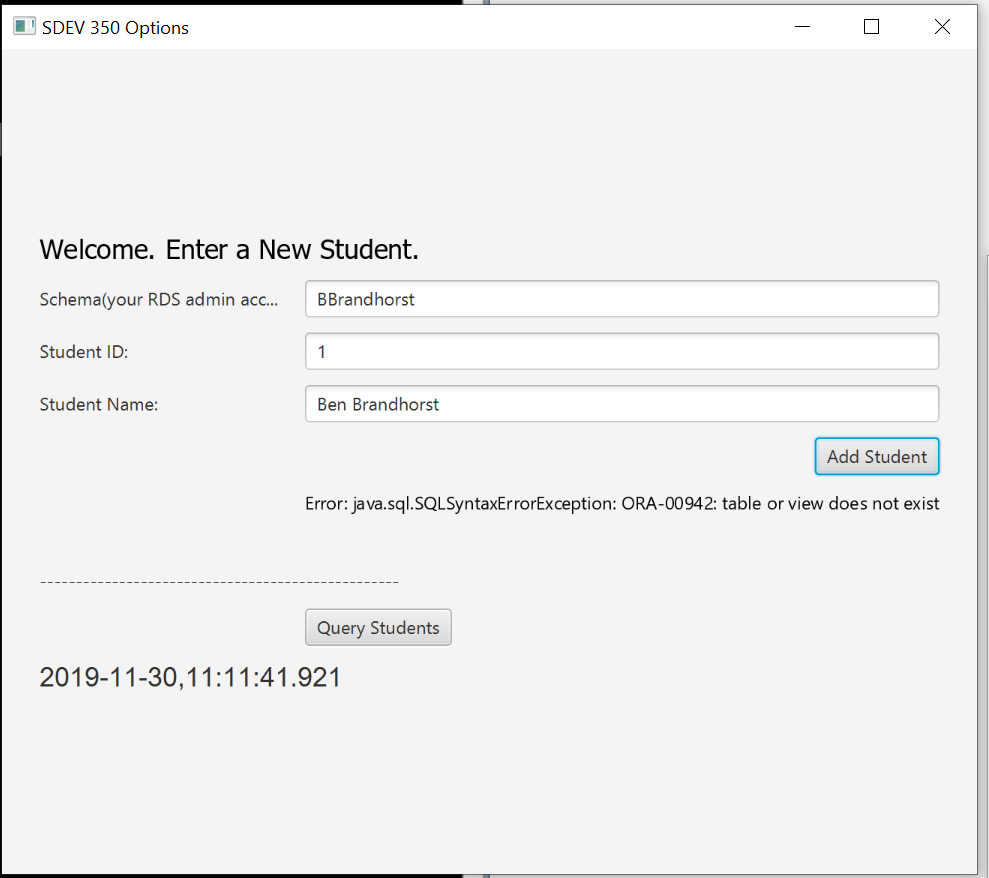
**Figure 1a – Launching the SDEV350App.jar**



**Figure 1b – Connecting to the Oracle Database by Selecting “Sign In”**

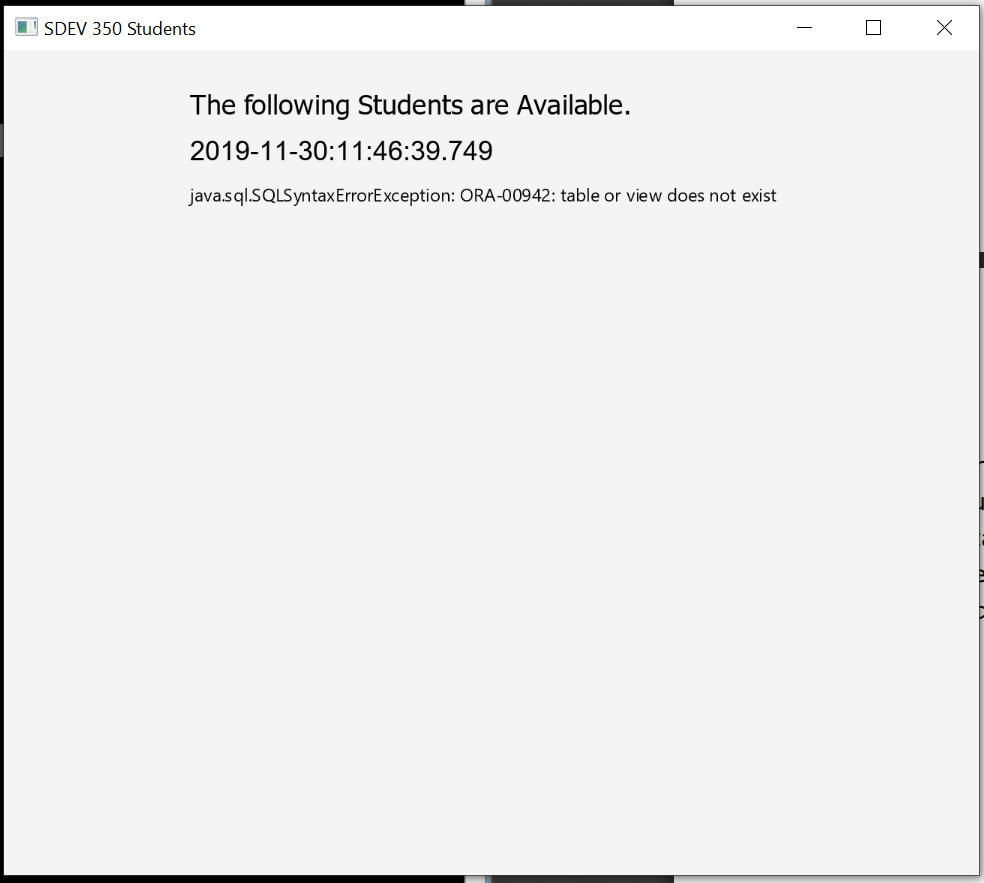


**Figure 1c – Entering a New Student Record and Selecting “Add Student”**

****

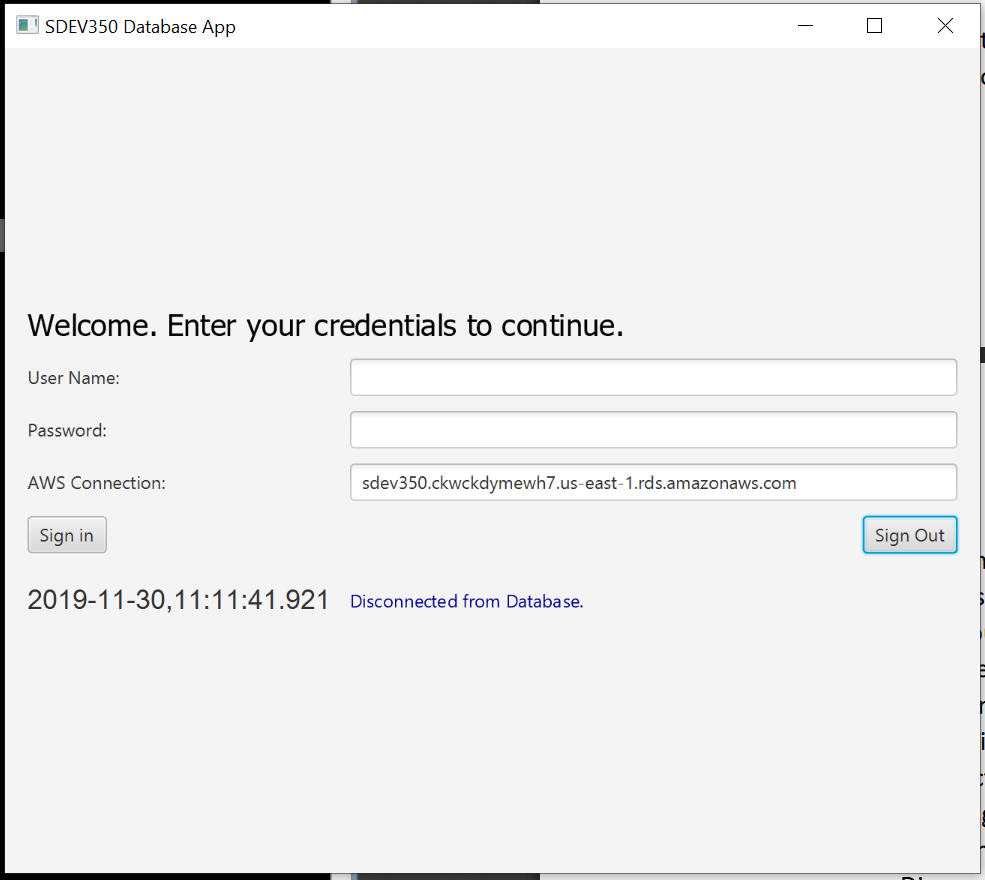
The directions here were confusing and I started this assignment late enough in the week to not really have time to wait on feedback. I could have done one of two things. If I logged in with the Admin account but did not create a Students table, I would get a similar error to the one shown here because there would exist no table named Students to add records to. However, when reading the Lab4 Clarifying steps, we were instructed to create the Students table before trying to add records to it. If I create the table and use the Admin account, there would be no errors and the records insert successfully. So, I instead created a Student user that did not have rights to insert records into the Students table. Because the user does not have rights to the table, the database will not confirm the existence of the table to them. This is why we receive the error shown above.

**Figure 1d – Querying the Students Table by Selecting “Query Students”**

****

Again, the user does not have the SELECT rights for the Students table. Because of this, the database will not confirm whether a table with that name exists.

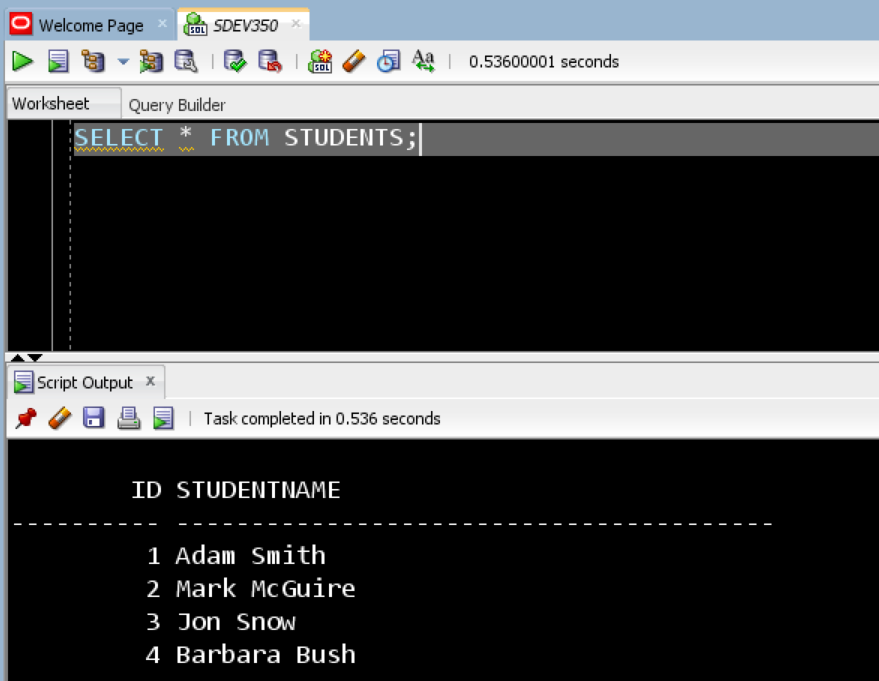
**Figure 1e – Disconnecting from the Oracle Database by Selecting “Sign Out”**

****

Here we see the results of the “Sign Out” button being clicked. The connection to the database is terminated and the Username and Password information is removed from the text boxes.

1. Using your assigned Student account, create a table named Students with two columns (ID (integer, primary key) and Name (varchar2(40)). Insert at least 4 records into the Students table. Document that you have successfully created the table and inserted at least 4 records by running a Query and capturing the results. (Note: If you have a previous Students table, you will need to delete that table to create the new replacement.)

**Figure 2 – Students Table Query**

****

1. Create a user namedTS1FirstnameLastname where Firstname and Lastname are your first and last name. The following specification should be associated with this user:

a. Provide the user a Profile you previously created.

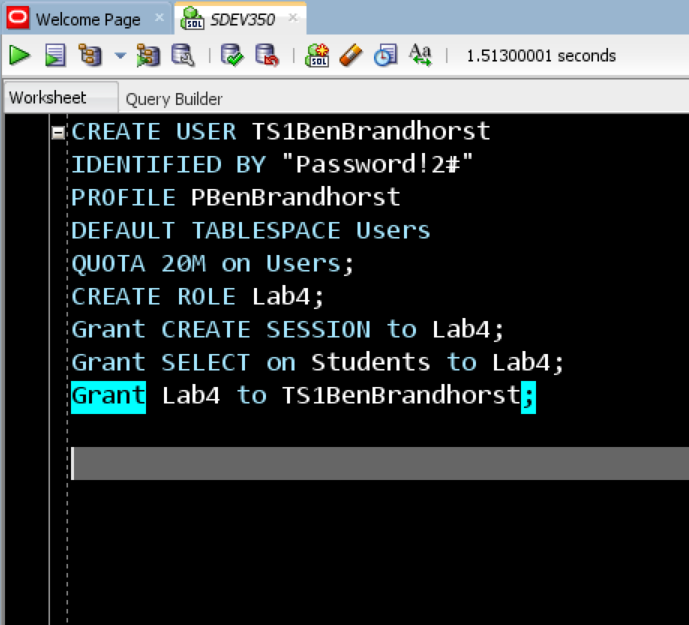
b. Assign the user to the Users Tablespace with a Quota of 20M.

c. Use a Role to provide this user permissions to Connect to the database, and Select from the Students Table in your admin account. The user should receive no more privileges. Name the Role TS1RoleFirstnameLastname.

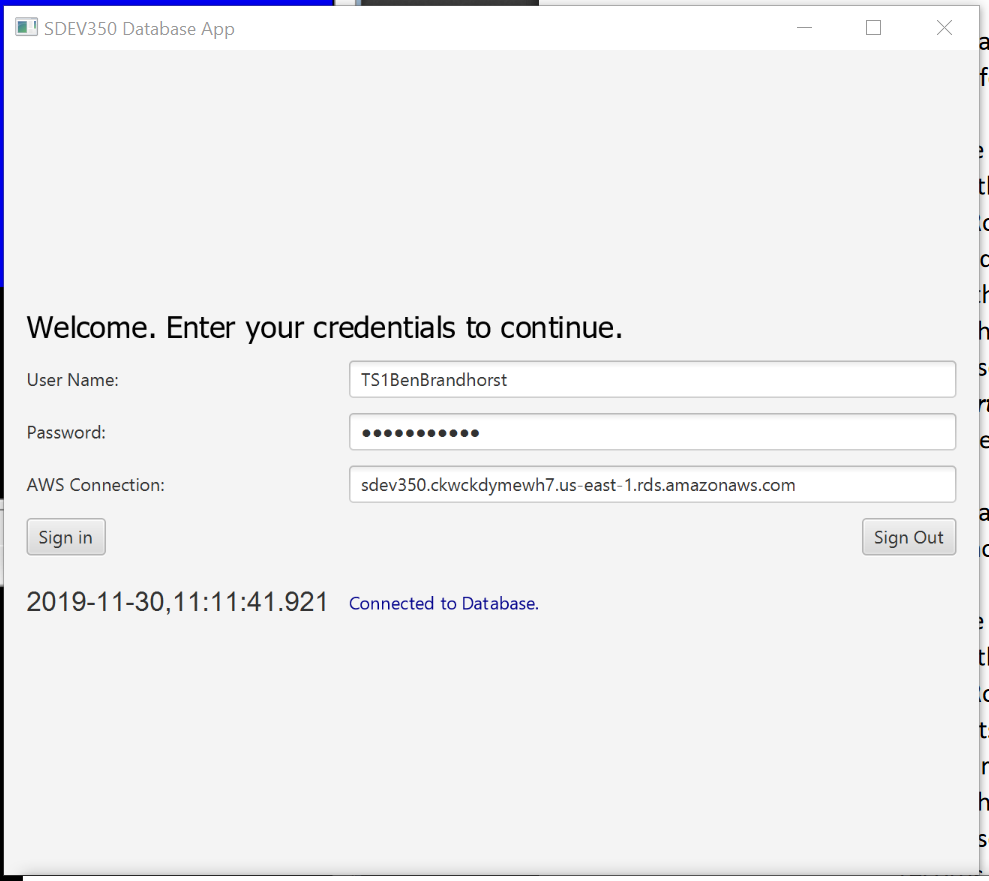
d. Using the SDEV350App.jar application and new user you just created, connect to the database and demonstrate you can Select from the Students table but you are notable to Insert new records.

e. Document and fully describe in the word document the results of this step

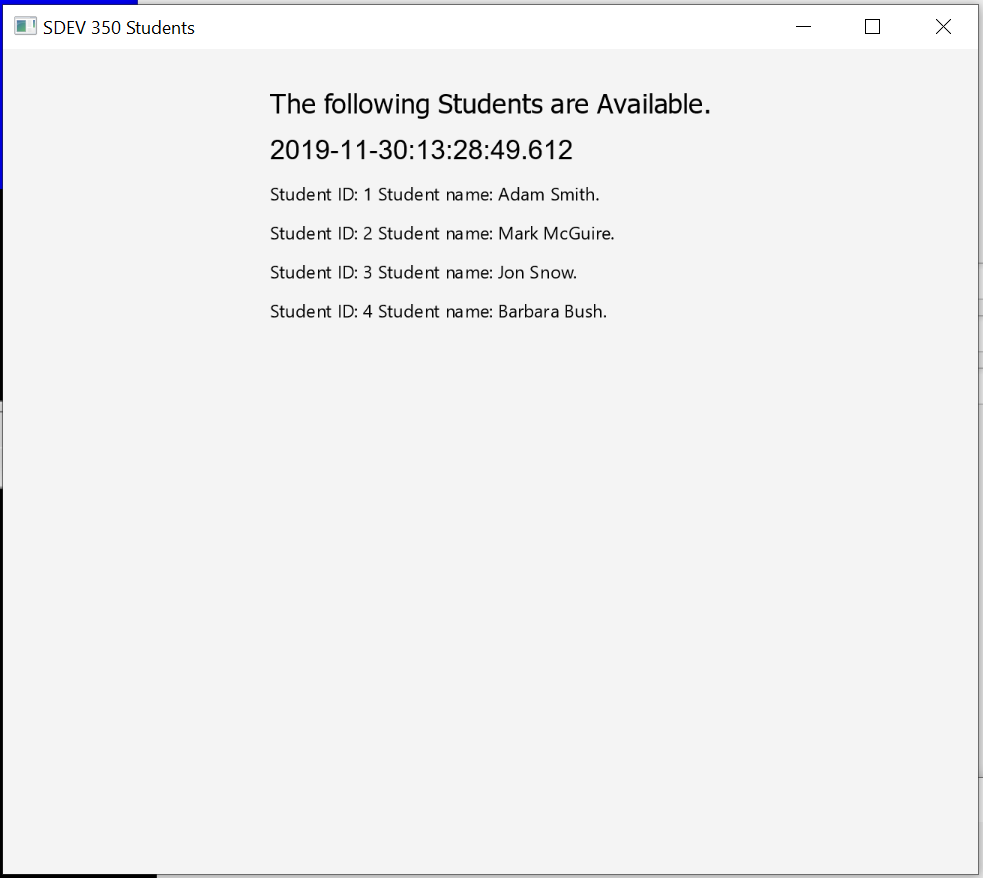
**Figure 3-1 Creating TS1BenBrandhorst**



**Figure 3-2 Connecting to the Database with newly created user**

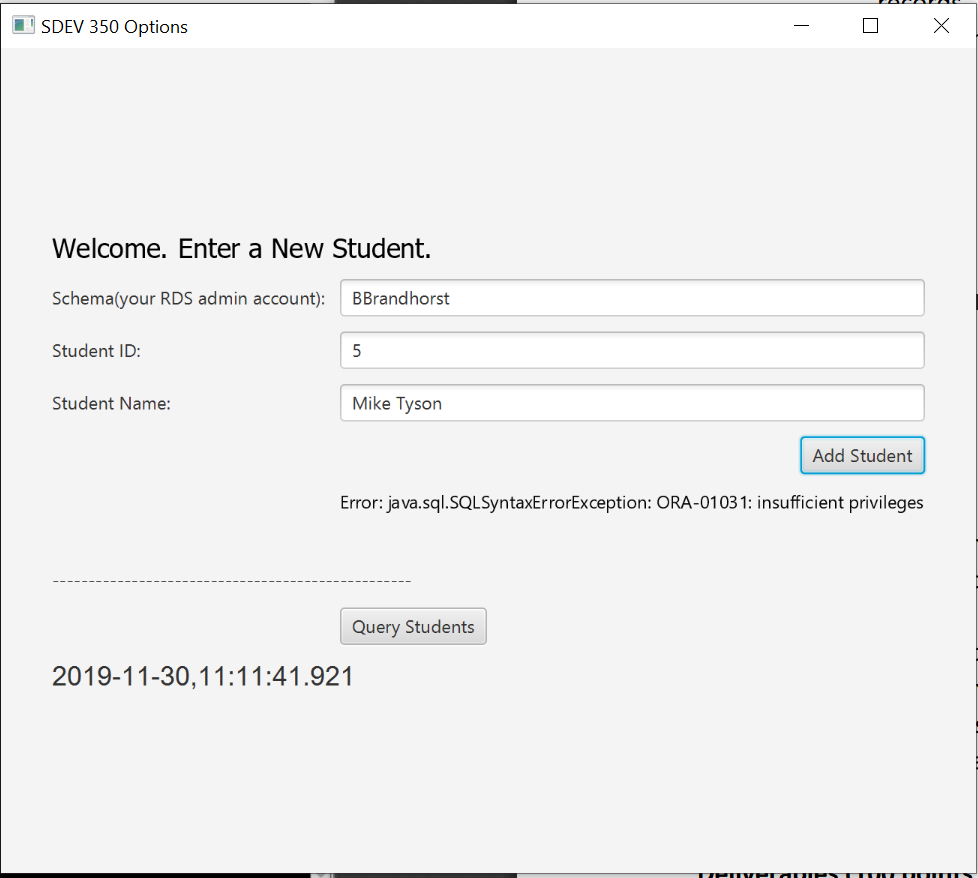
****

**Figure 3-3 – Selecting from Students Table**



Here we can see that the TS1BenBrandhorst user has the appropriate privileges to SELECT from the Students table. Because of this, the query run returns the records contained in the Students table.

**Figure 3-4 – Inserting New Records into Students Table**

****

This demonstrates that user TS1BenBrandhorst does not have the appropriate privileges to enter new records into the Students table. As a result, the user is unable to use the full functionality of the Java program.

1. Create a user named TS2FirstnameLastname where Firstname and Lastname. The following specification should be associated with this user:

a. Provide the user a Profile you previously created.

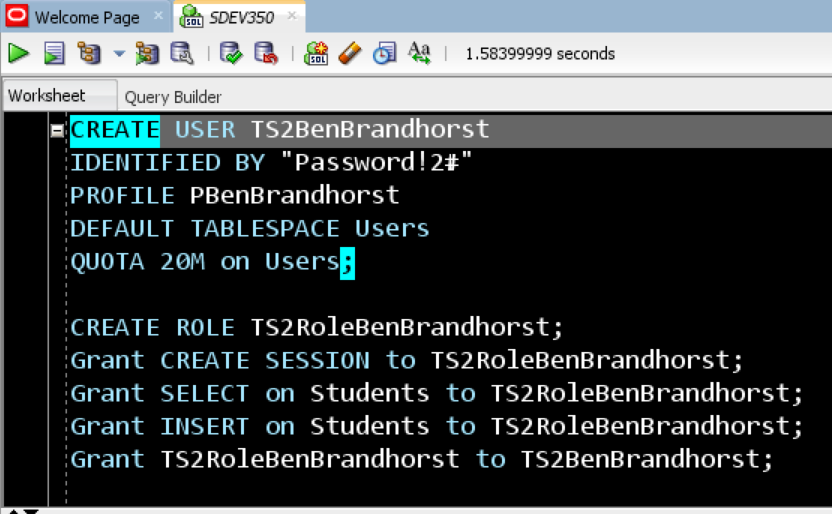
b. Assign the user to the Users Tablespace with a Quota of 20M.

c. Use a Role to provide this user permissions to Connect to the database, Select from the Students Table, and Insert records into the Students table in your account. The user should receive no more privileges. Name the Role TS2RoleFirstnameLastname.

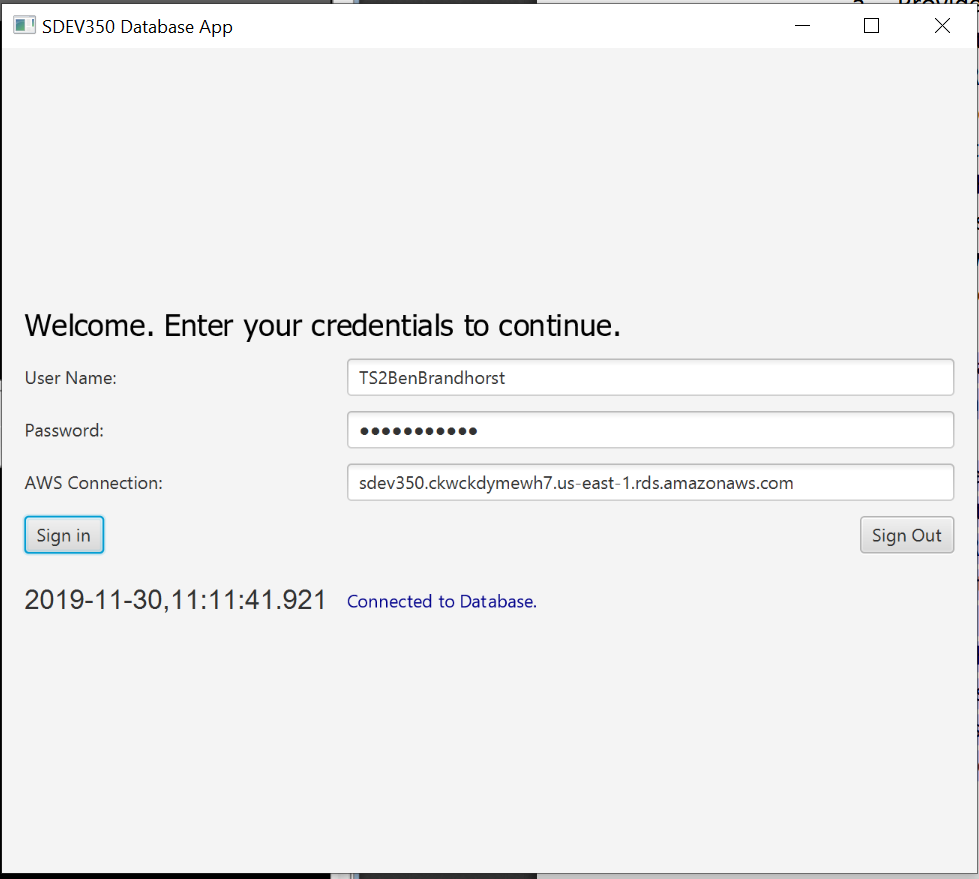
d. Using the SDEV350App.jar application and new user you just created, connect to the database and demonstrate you can Selectf rom the Students table and Insert new records.

e. Document and fully describe in the word document the results of this step.

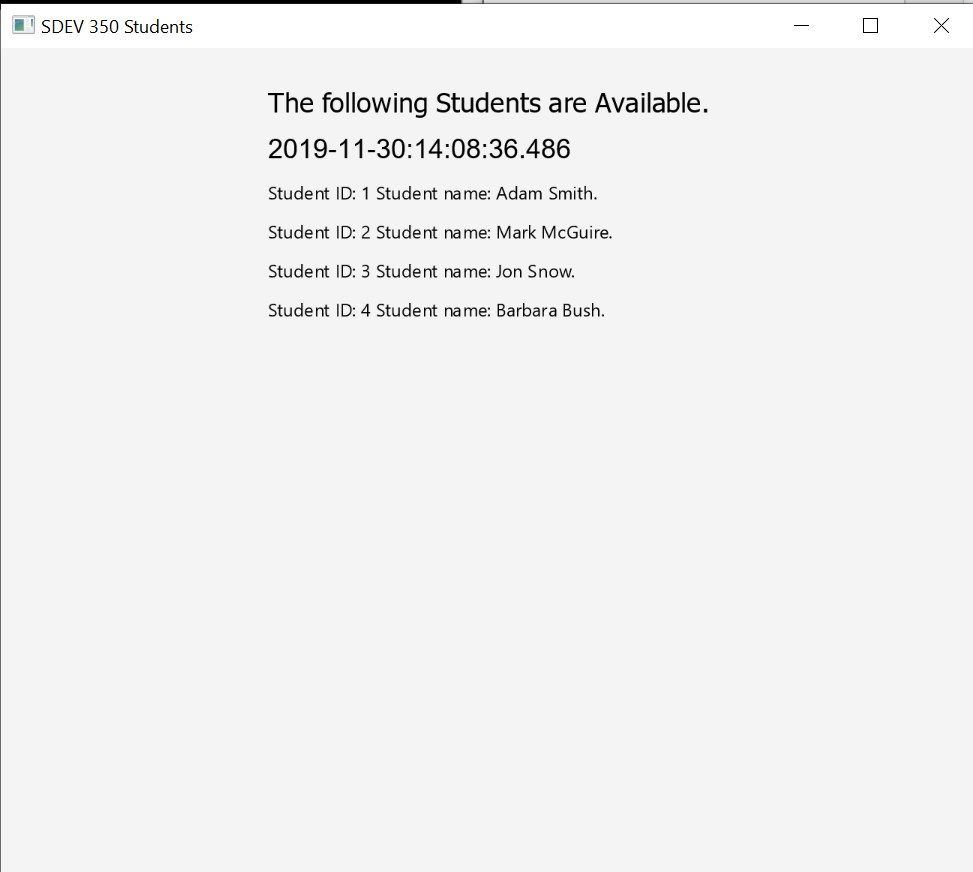
**Figure 4-1 – Creating TS2BenBrandhorst**

****

**Figure 4-2 – Connecting to the database with user TS2BenBrandhorst**

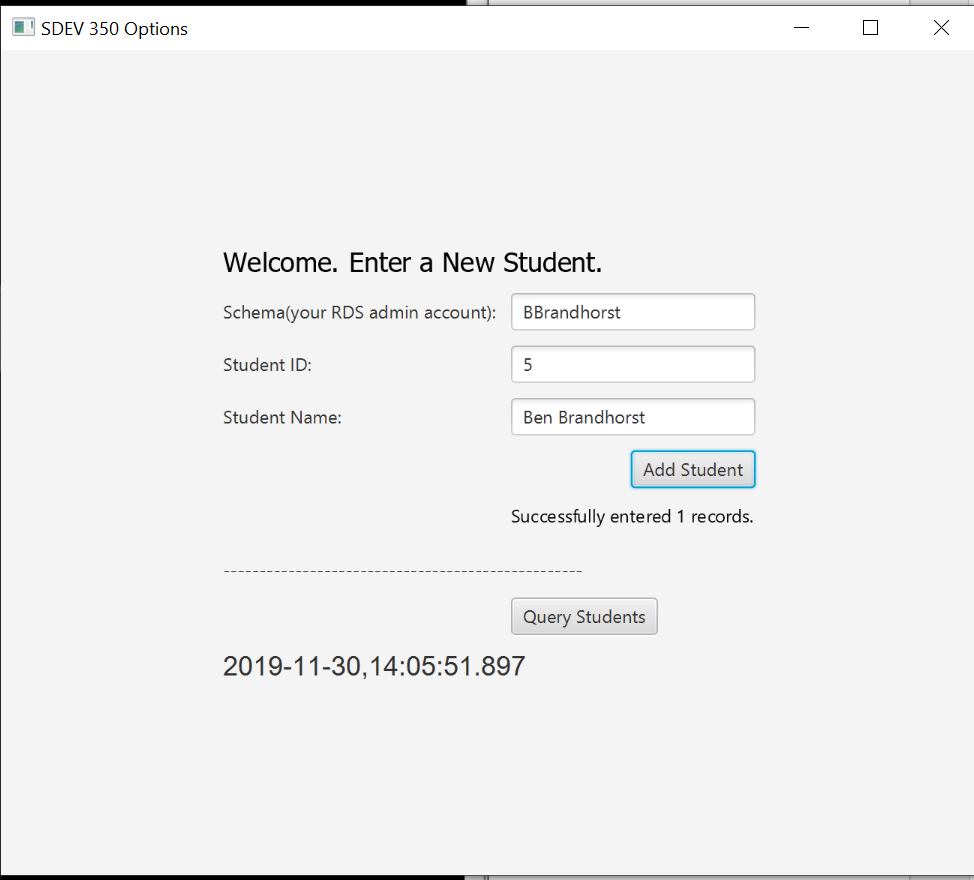
****

**Figure 4-3 – Selecting from Students Table**

****

Here we can see that the TS2BenBrandhorst user has the appropriate privileges to SELECT from the Students table. Because of this, the query run returns the records contained in the Students table.

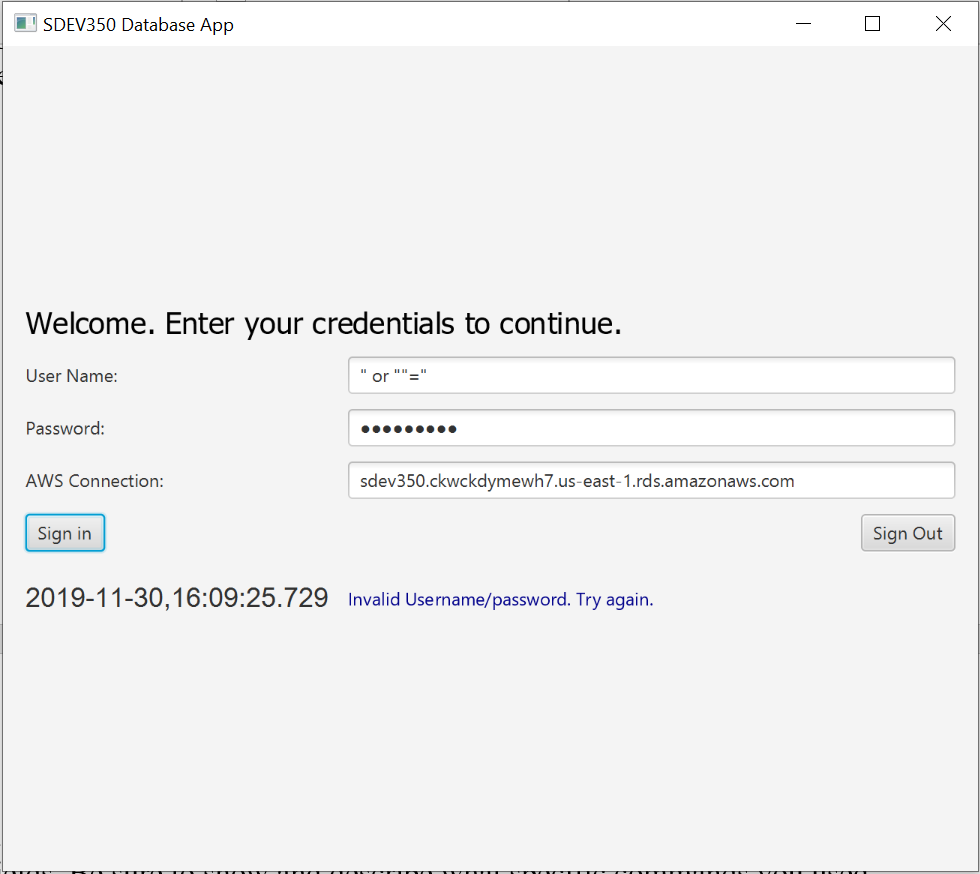
**Figure 4-4 – Inserting New Records into Students Table**

****

Here we see that user TS2BenBrandhorst has the appropriate privileges to insert records into the Students table. This is evidenced by the successful insertion of a new record into the table as demonstrated above.

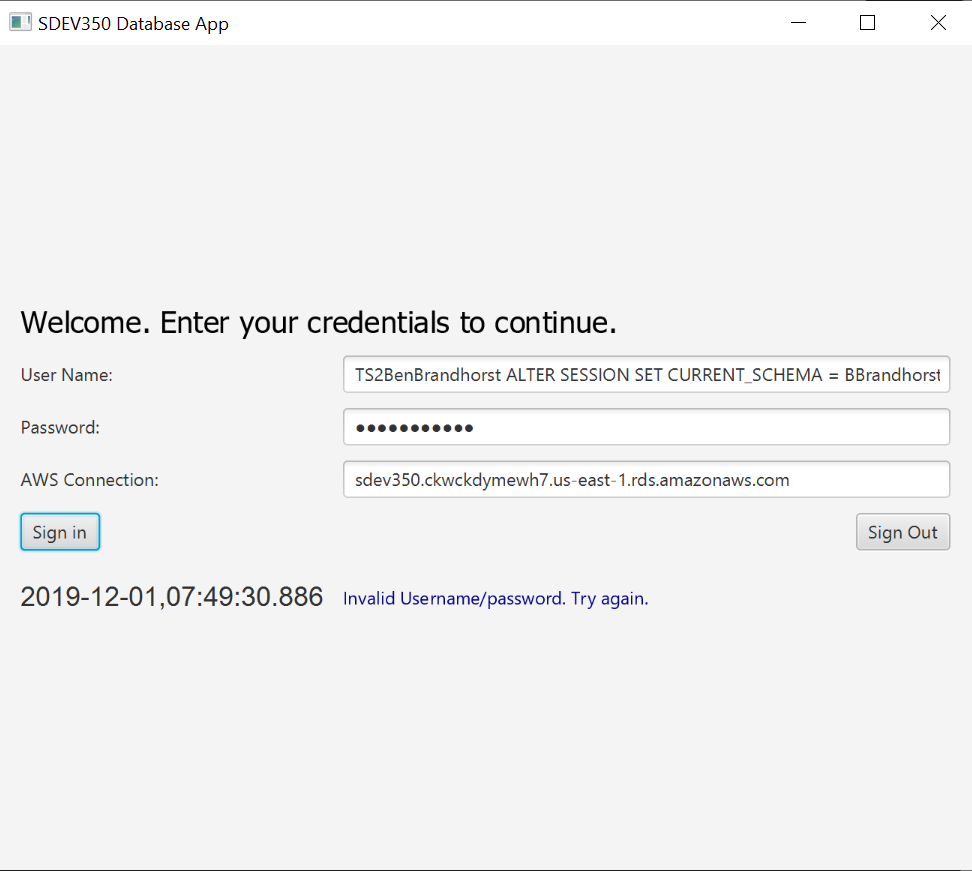
1. Using the details found in the reading this week, attempt to inject additional commands via SQL injection in the Insert fields. Be sure to show and describe what specific commands you used and why. Based on your results either make recommendations for improvement or provide details about the possible methods the application is using to protect itself against SQL injection. Be specific by providing actual Java code examples. Note: You should think like a hacker here. All you have is the interface to try to break into via SQL injection. You don’t have the code. So you have to test and see what will break and what information you can find out from trying different SQL injection attacks. Be sure to try SQL injection attacks for each text field that has user entry.

**Figure 5-1 – SQL Injection Attempt #1**

****

The above SQL injection attempt checks for input sanitization. With " or ""=" used for both the User Name and Password entries, if there were no sanitization taking place, it may be possible to access a list of users and passwords.

**Figure 5-2 – SQL Injection Attempt #2**



The above example where TS2BenBrandhorst ALTER SESSION SET CURRENT\_SCHEMA = BBrandhorst is used in place of the User Name was an attempt to login as TS2BenBrandhorst but access all of the privileges the Admin user has. Because neither injection attempt was successful, I suspect that this program is using parameterized SQL queries like something similar to

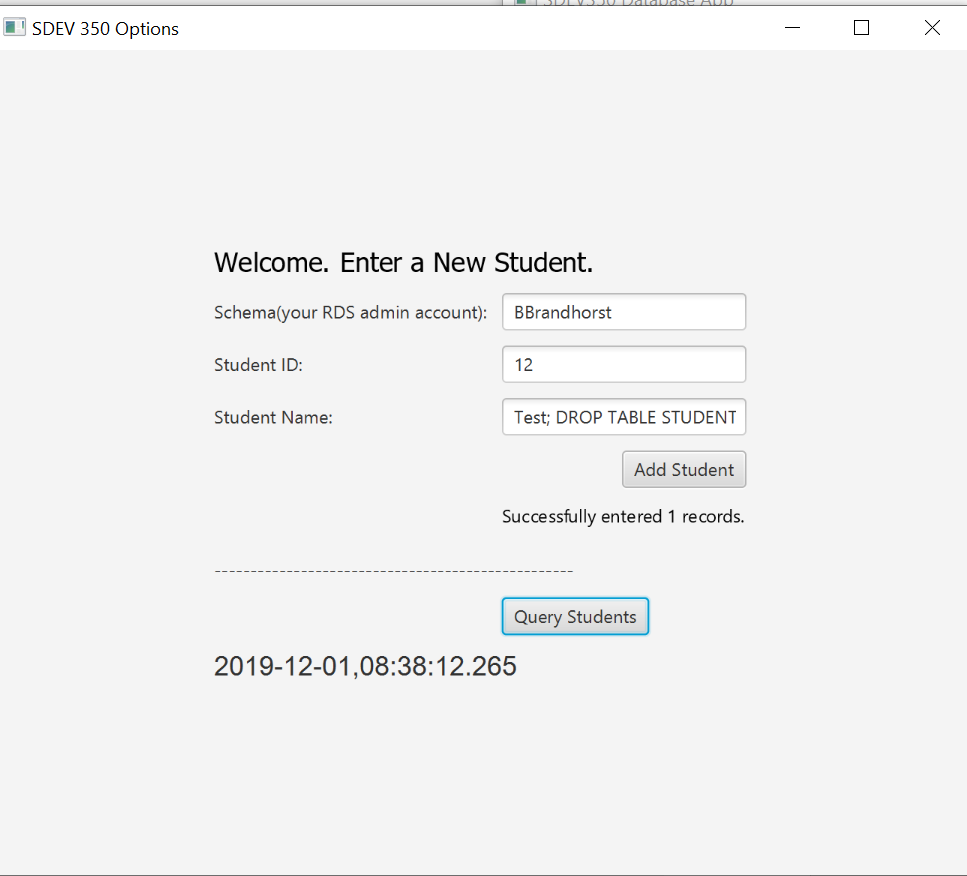
**PreparedStatement stmt = connection.prepareStatement("SELECT \* FROM users WHERE userid=? AND password=?");**

**stmt.setString(1, userid);**

**stmt.setString(2, password);**

**ResultSet rs = stmt.executeQuery();**

**Figure 5-3 – SQL Injection Attempt #3**



Here I attempted to add an additional SQL statement at the end of the student name. The actual input was Test; DROP TABLE STUDENTS. The response from the application was to create a new record with a Student Name of Test; DROP TABLE STUDENTS. Due to the fact that the semi colon went through I suspect this application is using prepared statements to sanitize user input. Like:

**String schema = schemaName.getText  
String query = “INSERT into” + schema + ”.STUDENTS (ID, Name) values (?, ?)”;**

**Recommendations for improvement.**

While I believe this application is fairly secure, there are a couple of recommendations I have for improvements to its design.

1. Using whitelist input validation as a secondary defense. This can be used as a second line of defense to mitigate against unwanted input types if the parameterized and prepared statements somehow fail to prevent unwanted commands from being passed.
2. Create and manage the table being modified with a user other than the Admin account. That way the schema entered is not the admin account name.