Lesson 01 Demo 07

Creating Stored Procedures

Objective: To explore stored procedures and execute them using the JDBC API called **callableStatement** for efficient and reusable database operations. This approach enables the direct invocation of stored procedures defined in the database from a Java application

Tool required: Eclipse IDE

Prerequisites: None

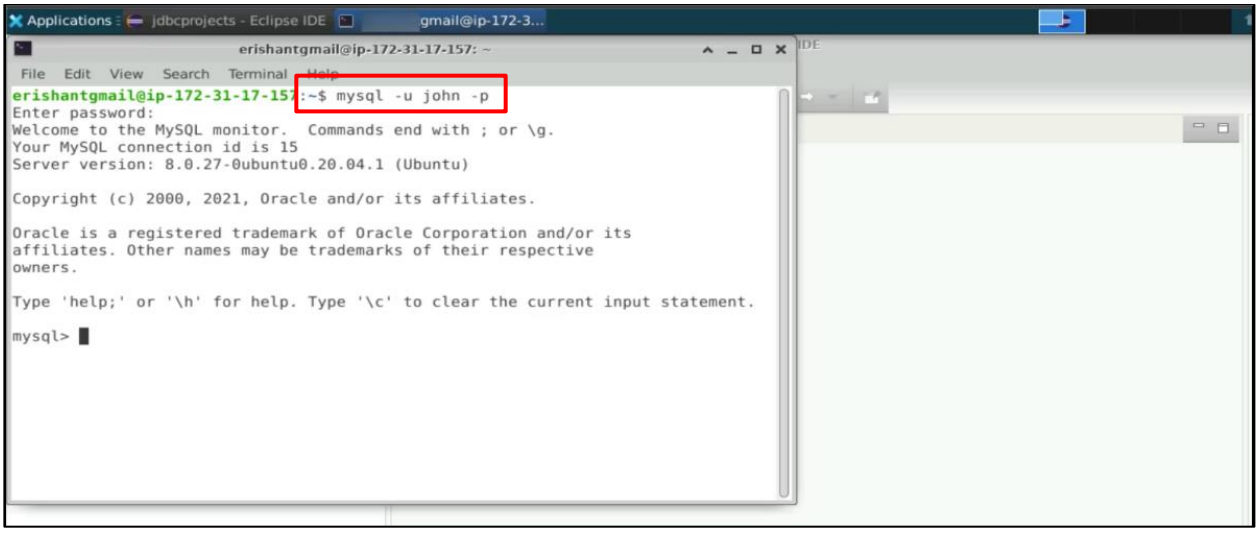
Steps to be followed:

1. Create a new table
2. Create a stored procedure
3. Create a method to execute a stored procedure
4. Create a stored procedure to read data
5. Create a method to call a stored procedure to read data

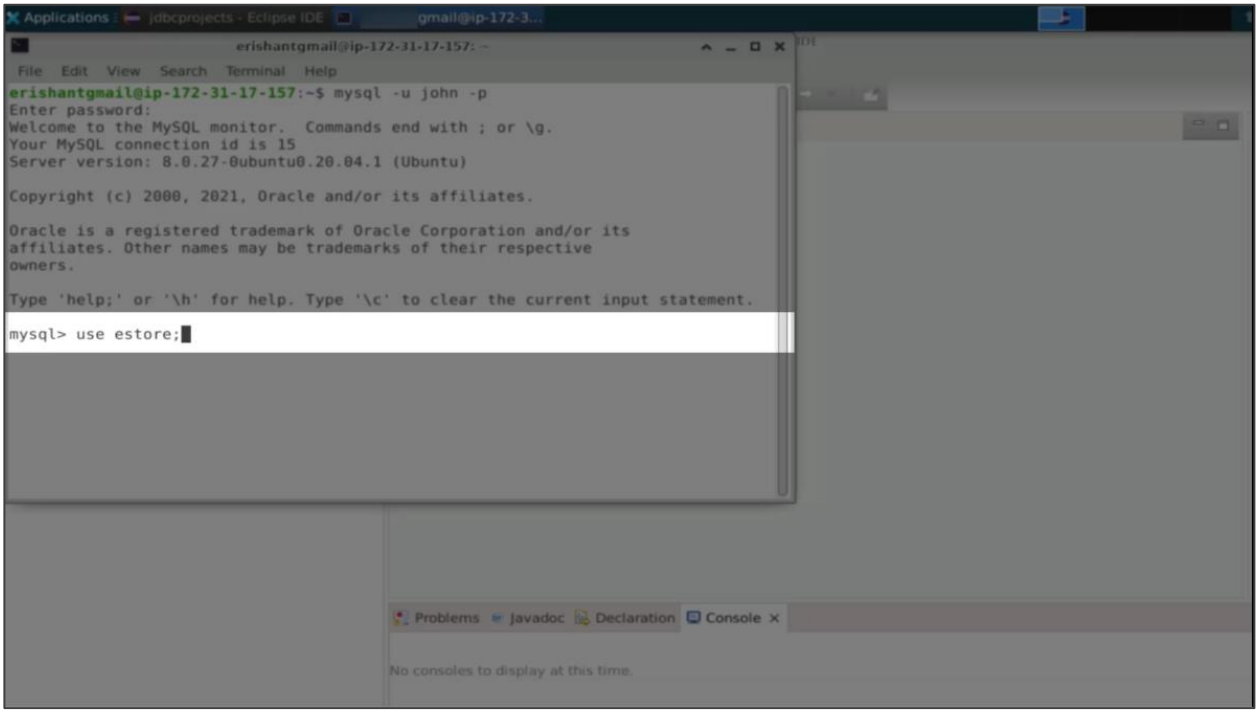
Step 1: Create a new table

1.1 Open the terminal, and log in to MySQL

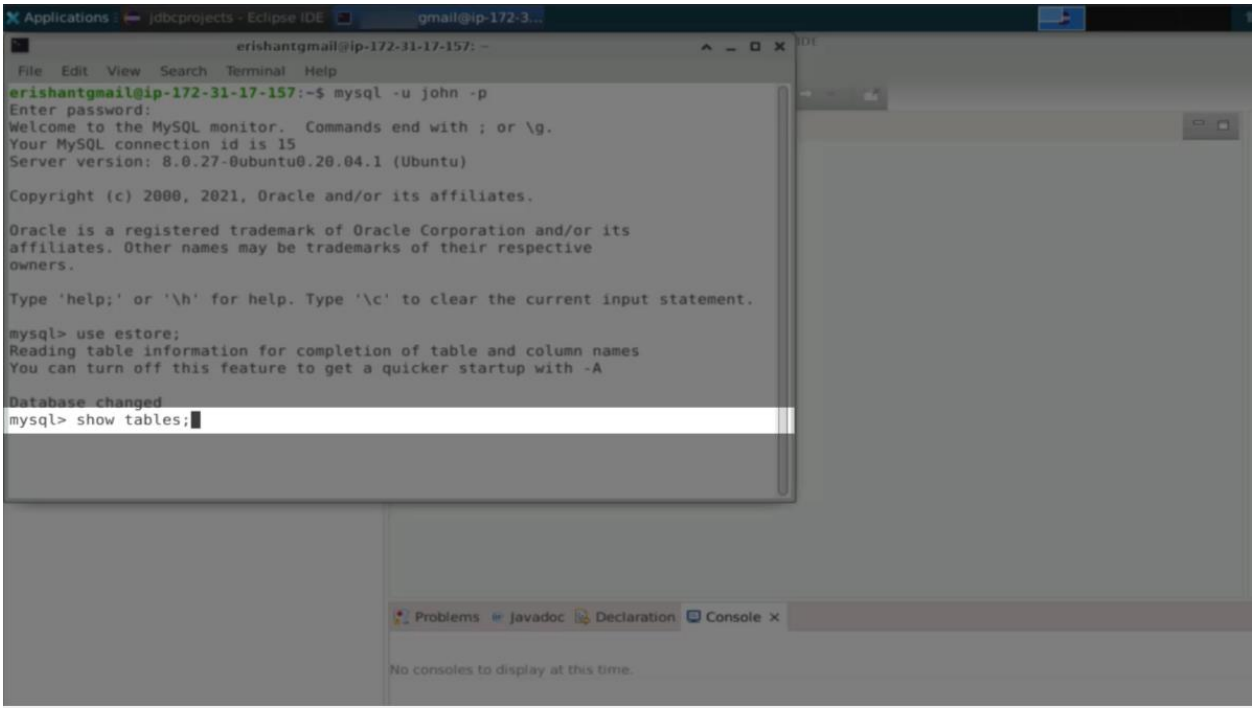
mysql –u john -p

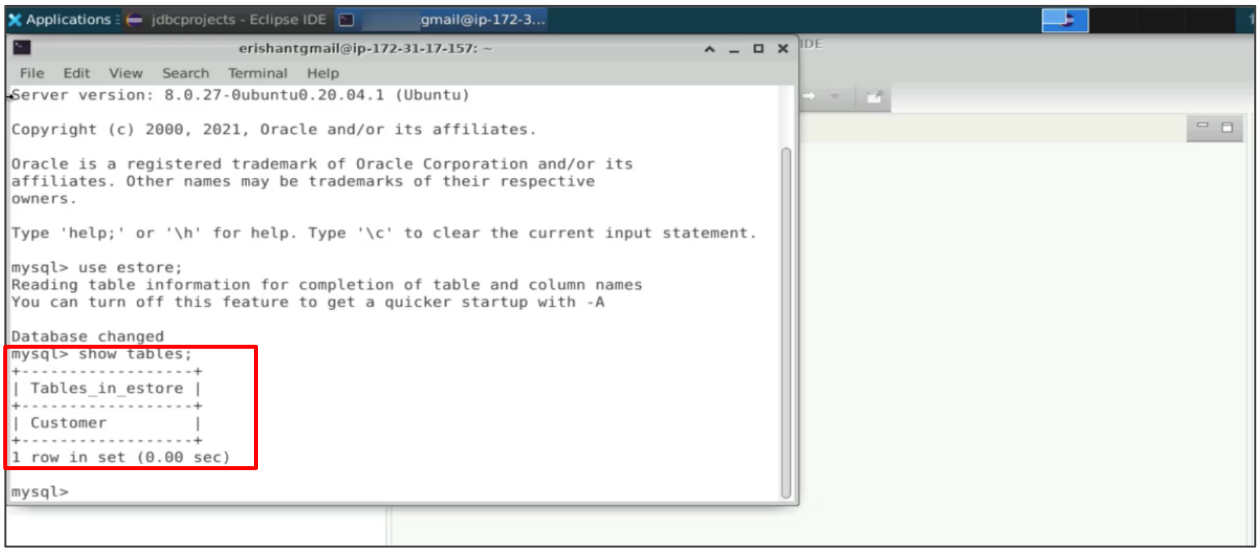


1.2 Use the command **use estore;** to change the database

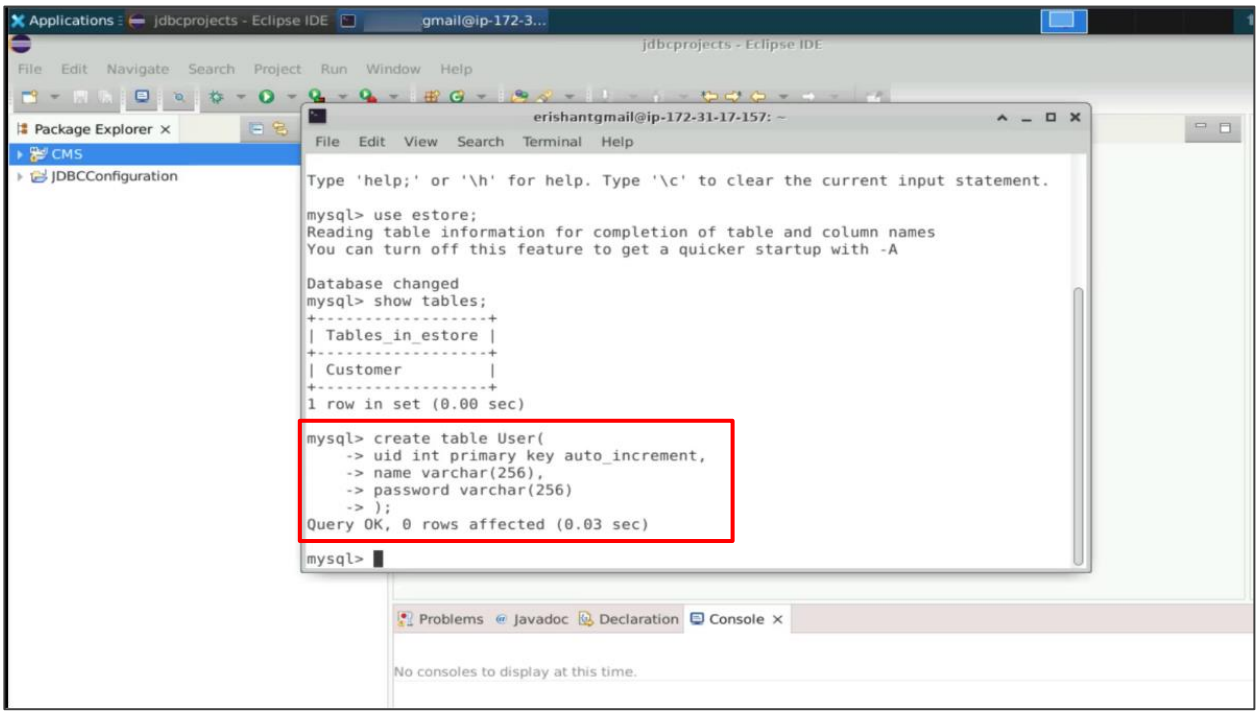


1.3 Use the **show tables;** command to check all tables available in the **estore** database

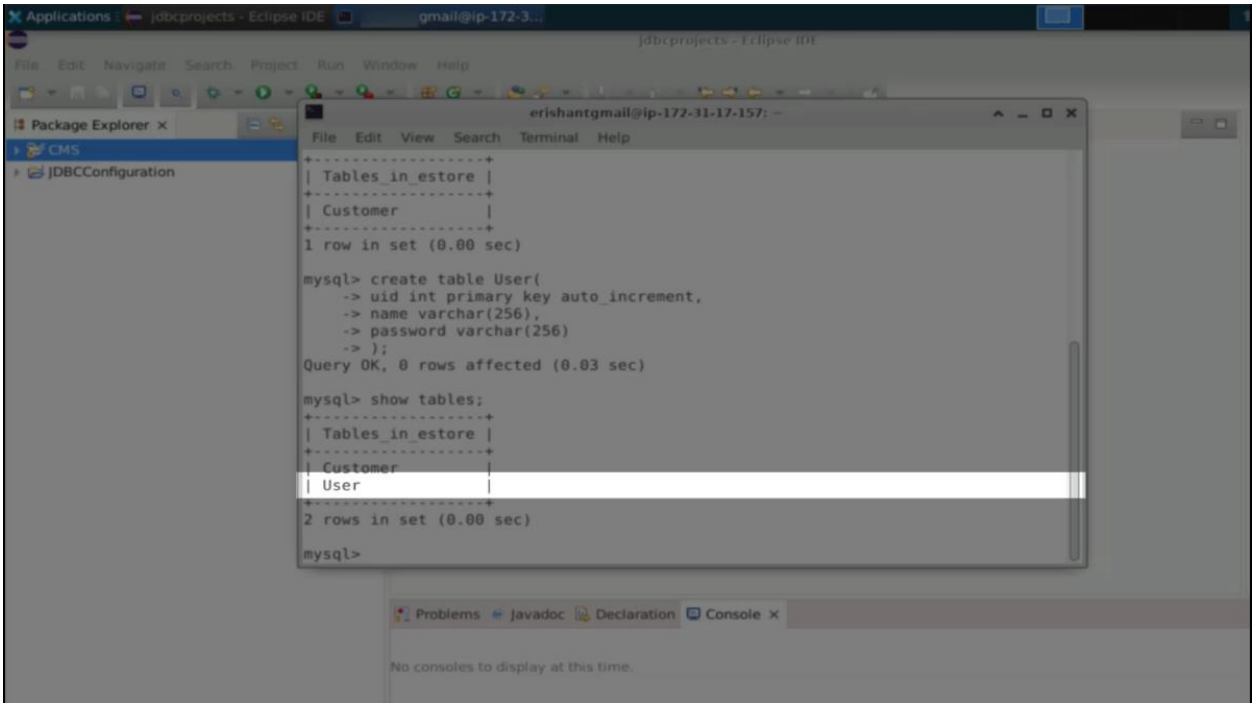




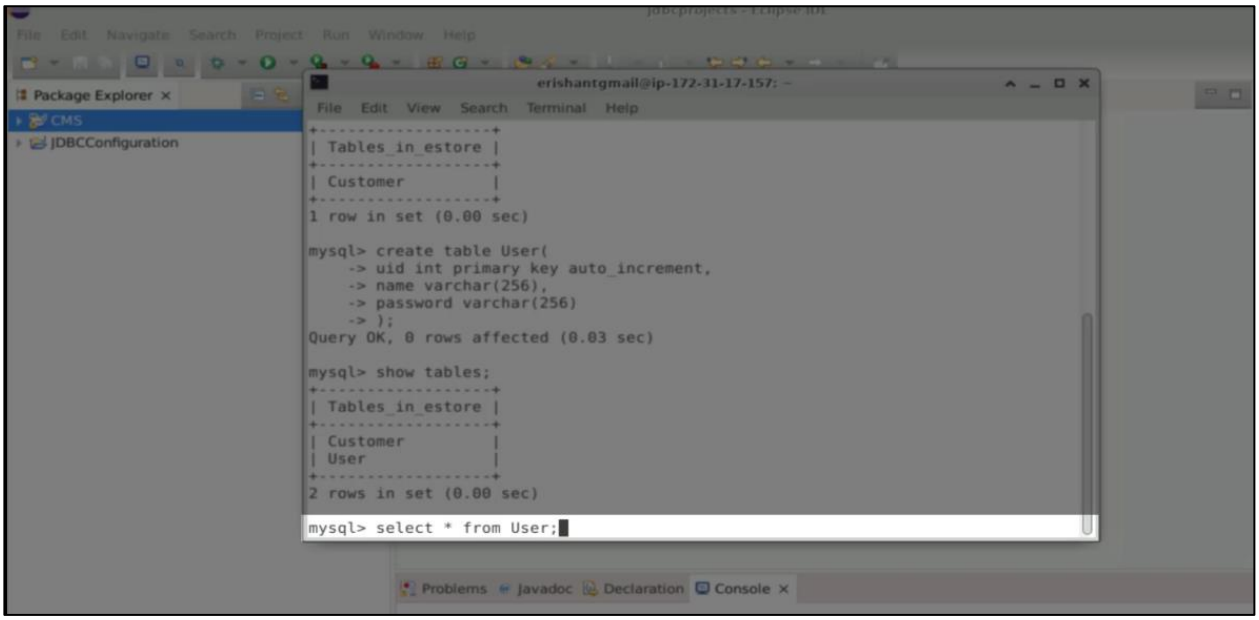
1.4 Use the **create table User();** command to create a new user table with attributes

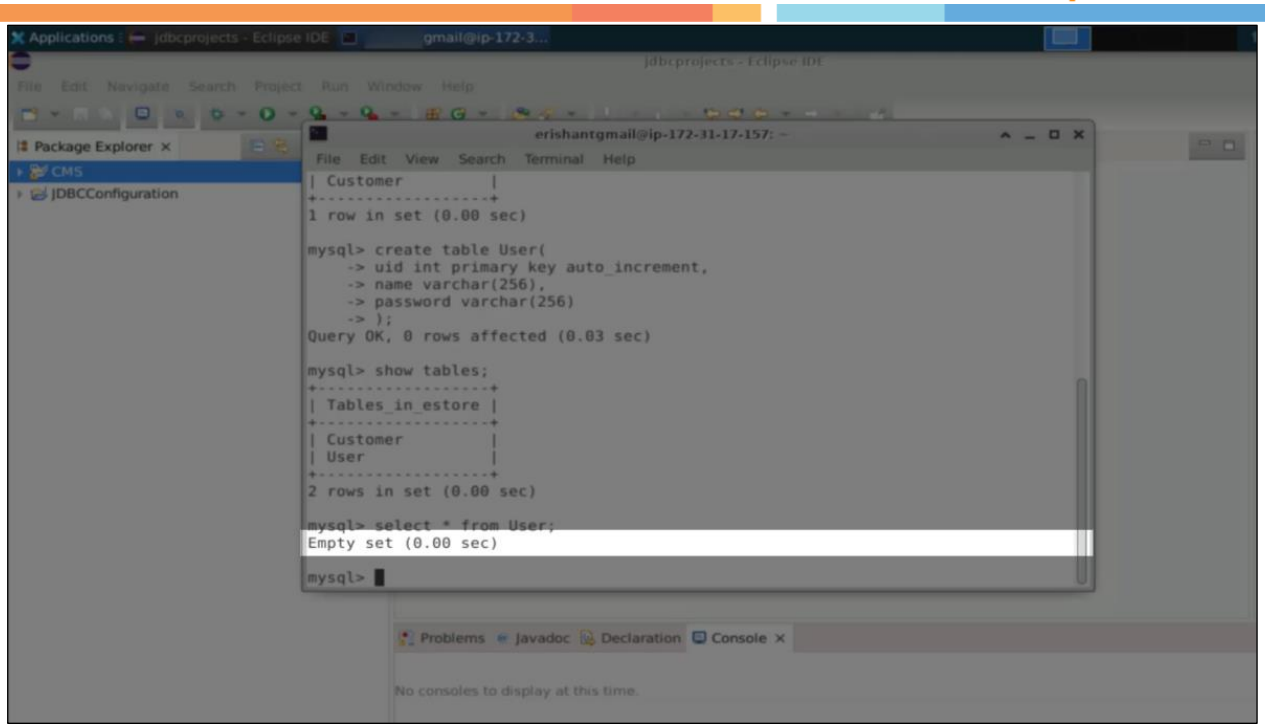


1.5 Write the command **show tables;** . The User table is created successfully and listed in the tables list.



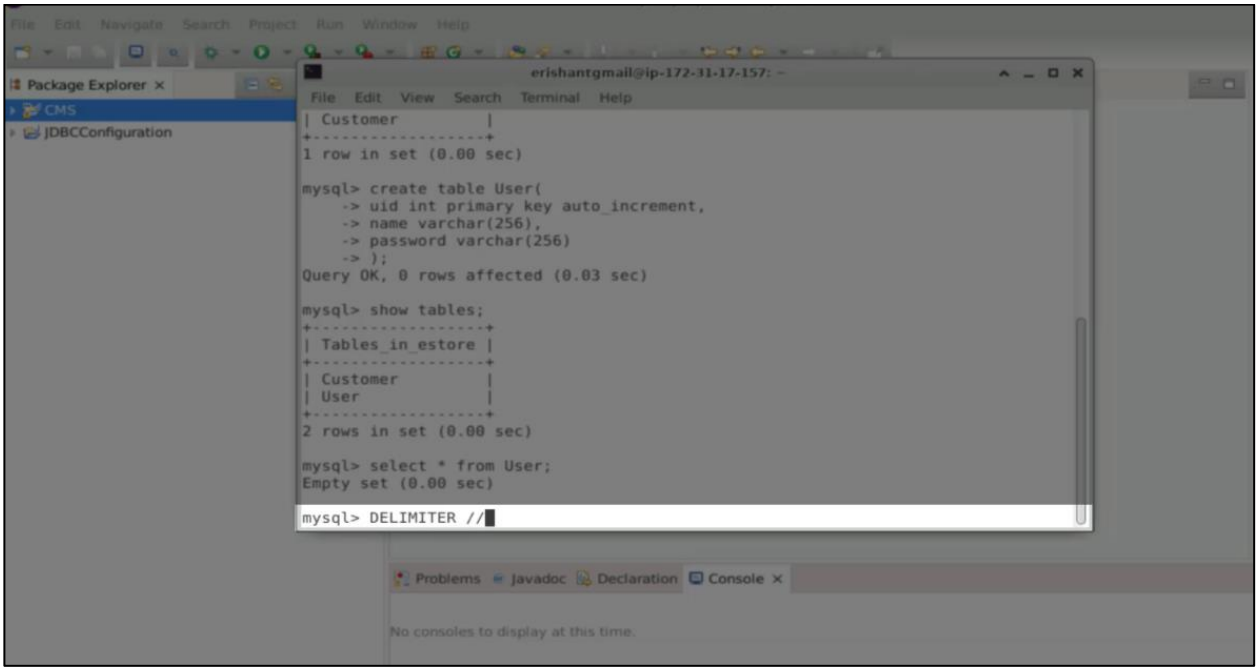
1.6 Use the **select * from User** command to select data from the User table. You can see that the table is empty for now.



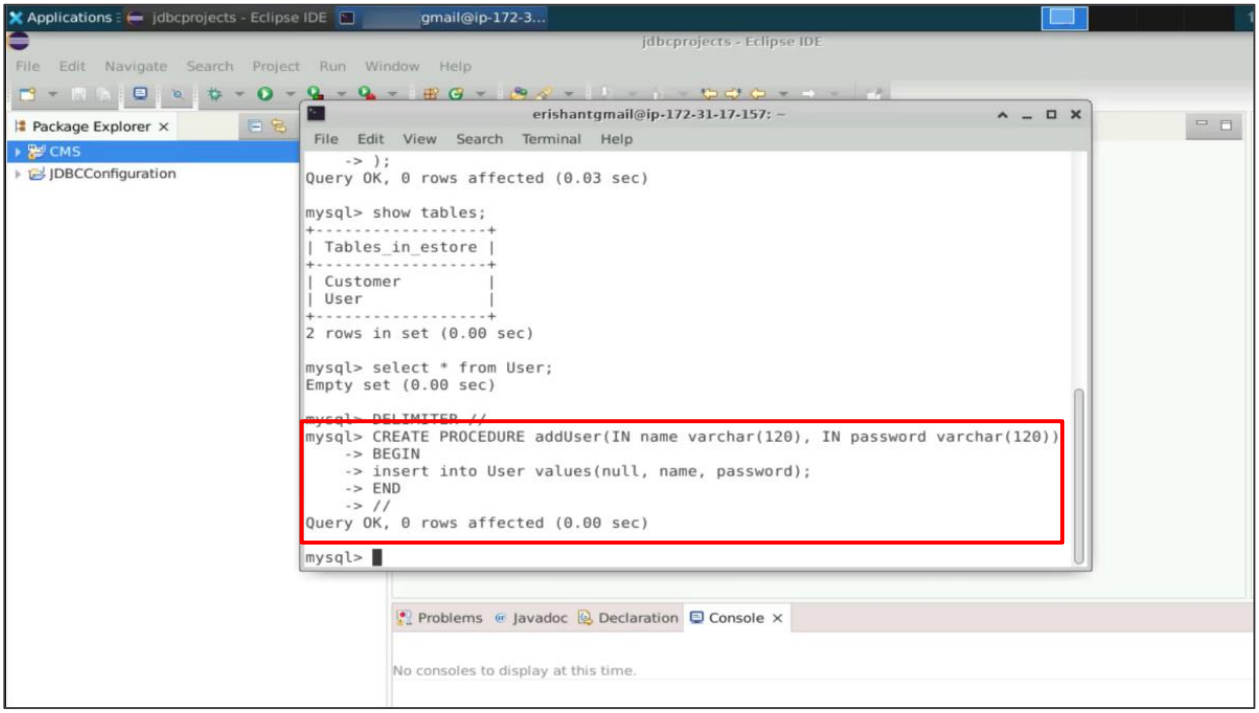

Applications
jdbcprojects - Eclipse IDE
gmail@ip-172-3...
jdbcprojects - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
CMS
JDBCConfiguration
erishantgmail@ip-172-31-17-157: ~
File Edit View Search Terminal Help
| Customer |
+----------+
1 row in set (0.00 sec)
mysql> create table User(
-> uid int primary key auto_increment,
-> name varchar(256),
-> password varchar(256)
->);
Query OK, 0 rows affected (0.03 sec)
mysql> show tables;
+----------+
| Tables_in_estore |
+----------+
| Customer |
| User |
+----------+
2 rows in set (0.00 sec)
mysql> select * from User;
Empty set (0.00 sec)
mysql>
Problems Javadoc Declaration Console
No consoles to display at this time.

Step 2: Create a stored procedure

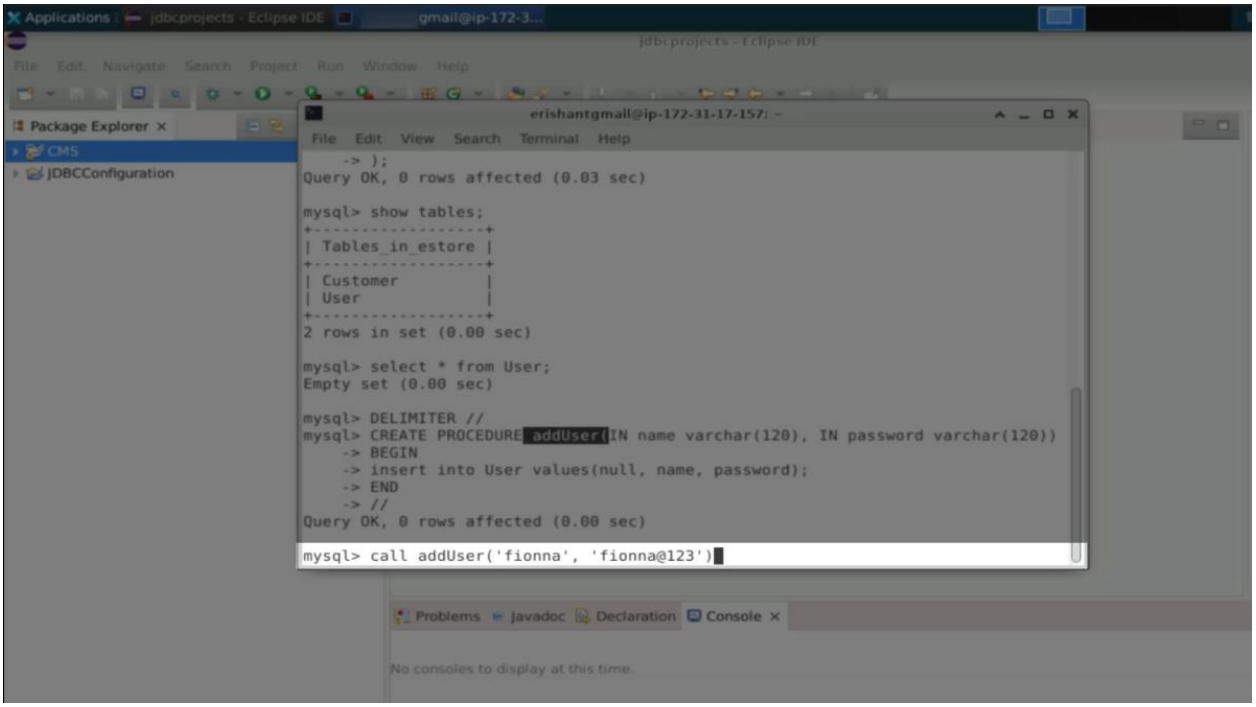
2.1 To enter records in this empty set, we must create a Stored Procedure. Change the **Delimiter** to **//** so it does not get confused with the **;** used while creating the procedure.



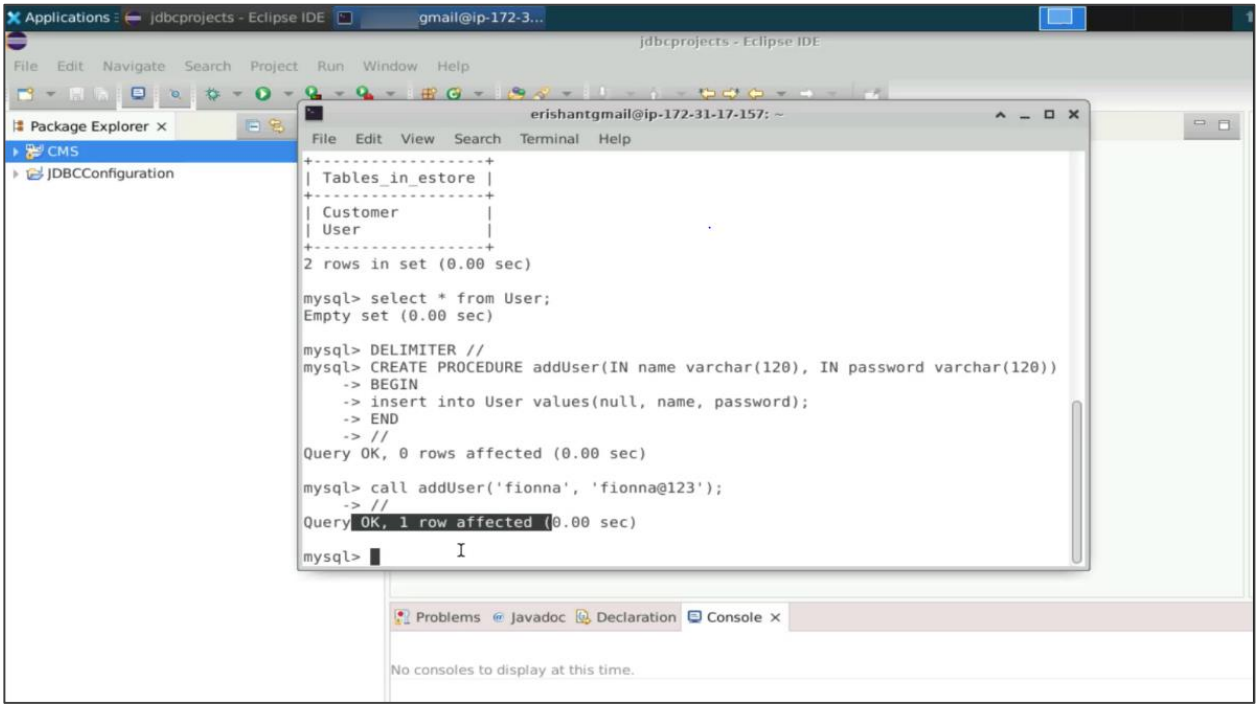
2.2 Run the **CREATE PROCEDURE addUser()** command to create a procedure with attributes



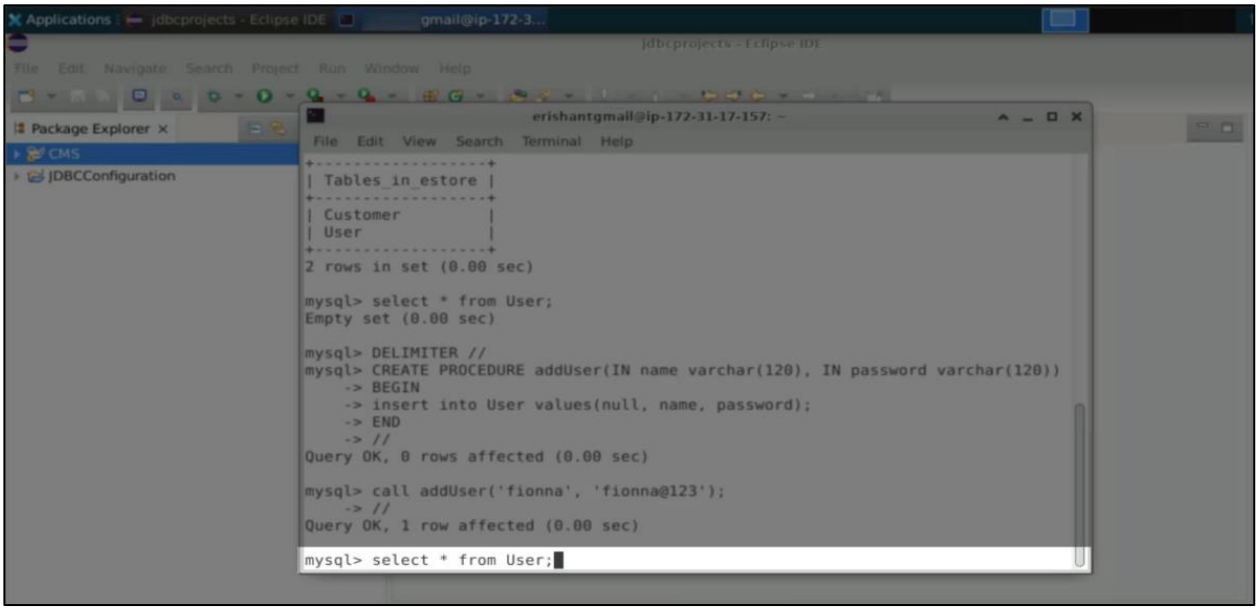
2.3 Execute the procedure we created by writing the **call addUser();** command with some data



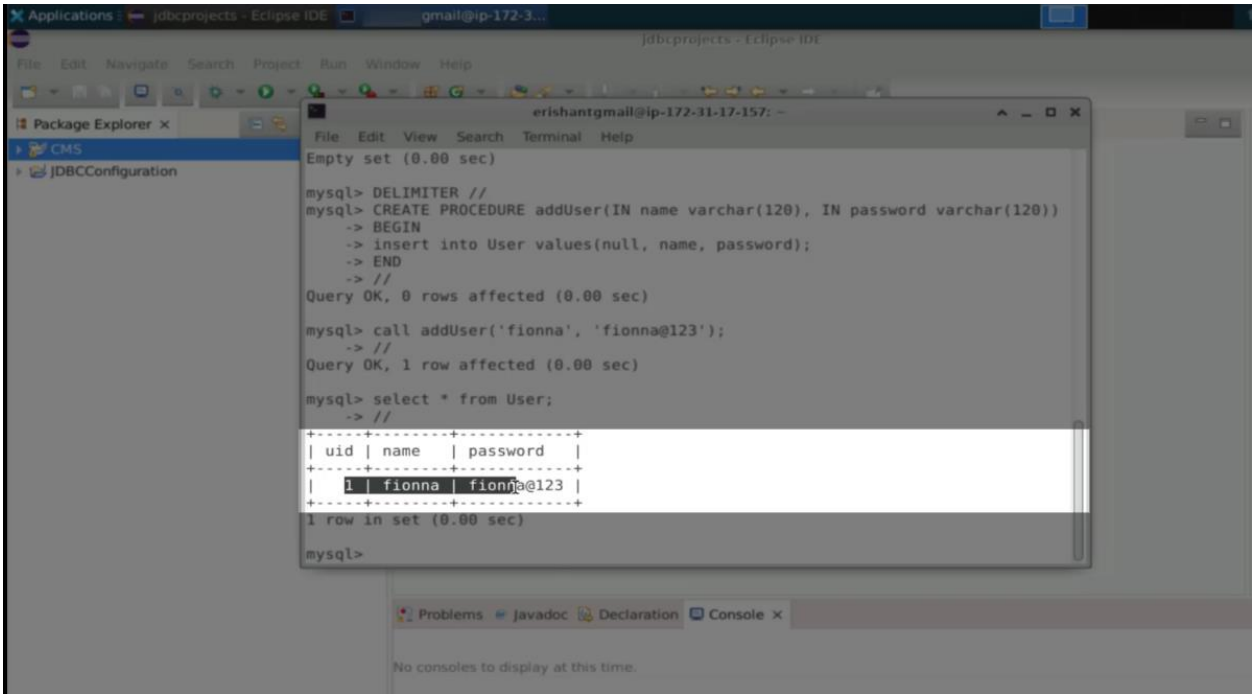
The command is executed, and you can see **1 row affected** as output.



2.4 Run the **select * from User** command to check if the data inserted in the table is successful or not

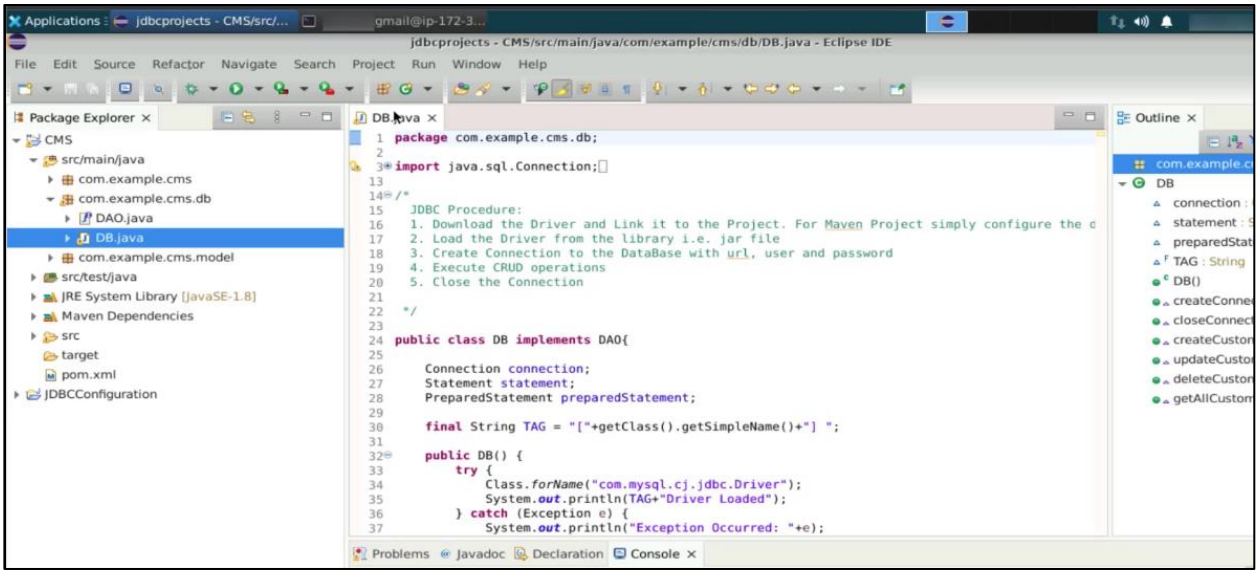


The record has been inserted successfully, as seen in the table.

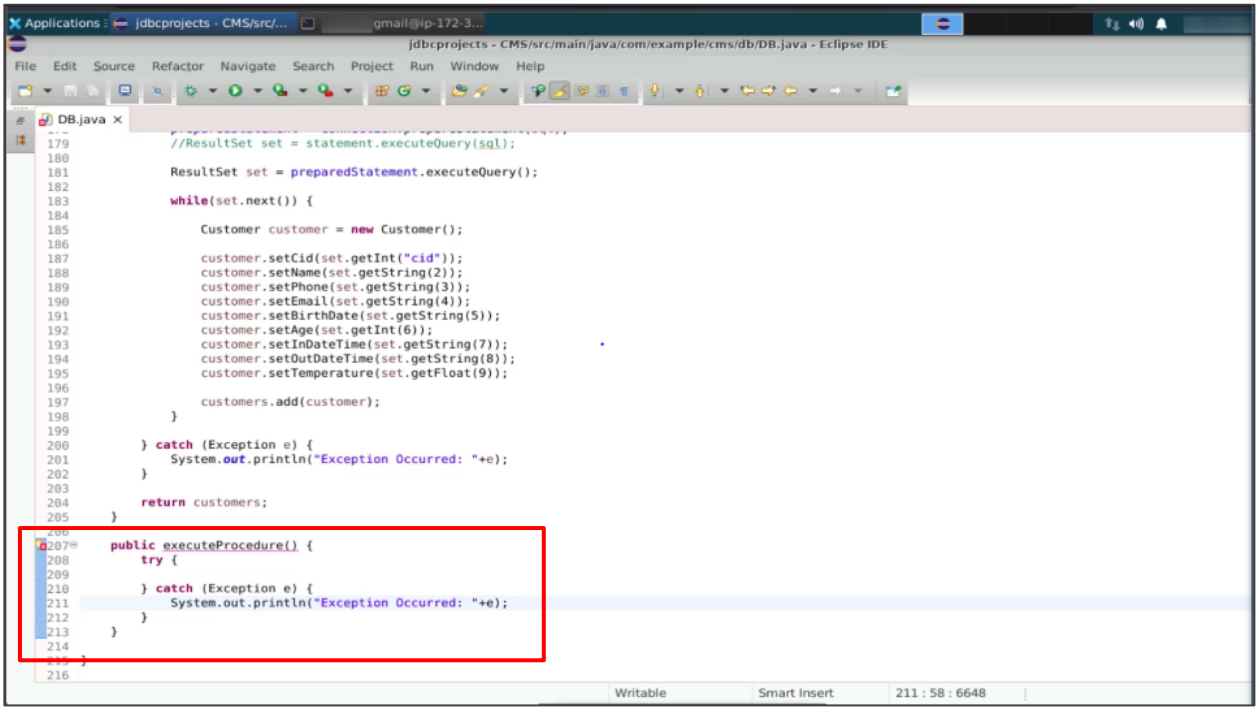


Step 3: Create a method to execute the stored procedure

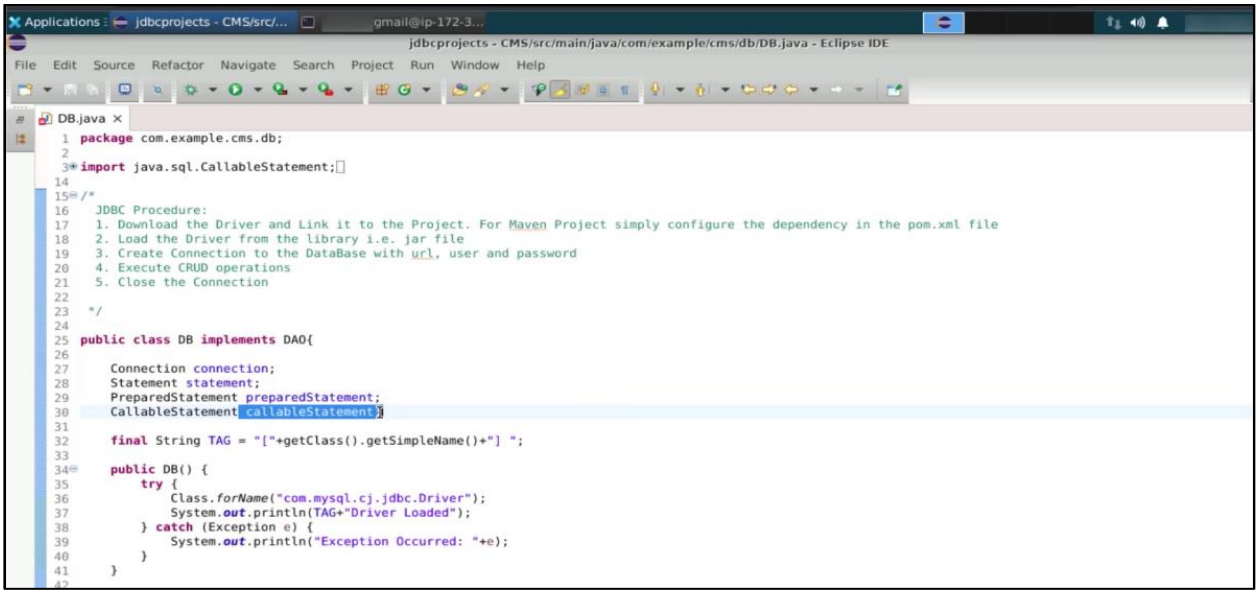
3.1 Open **Eclipse IDE** and go to the **DB.java** file



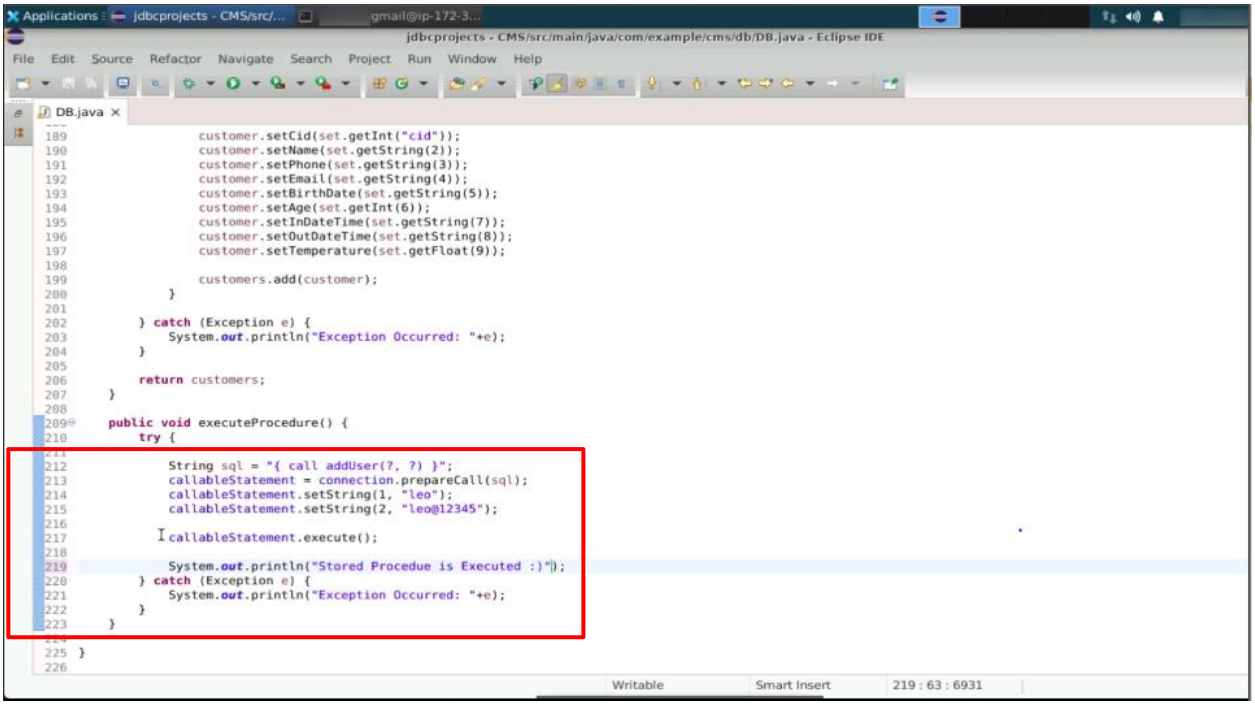
3.2 Execute the stored procedure, create a method **executeProcedure()** and put it inside the **try-catch** block to handle exceptions



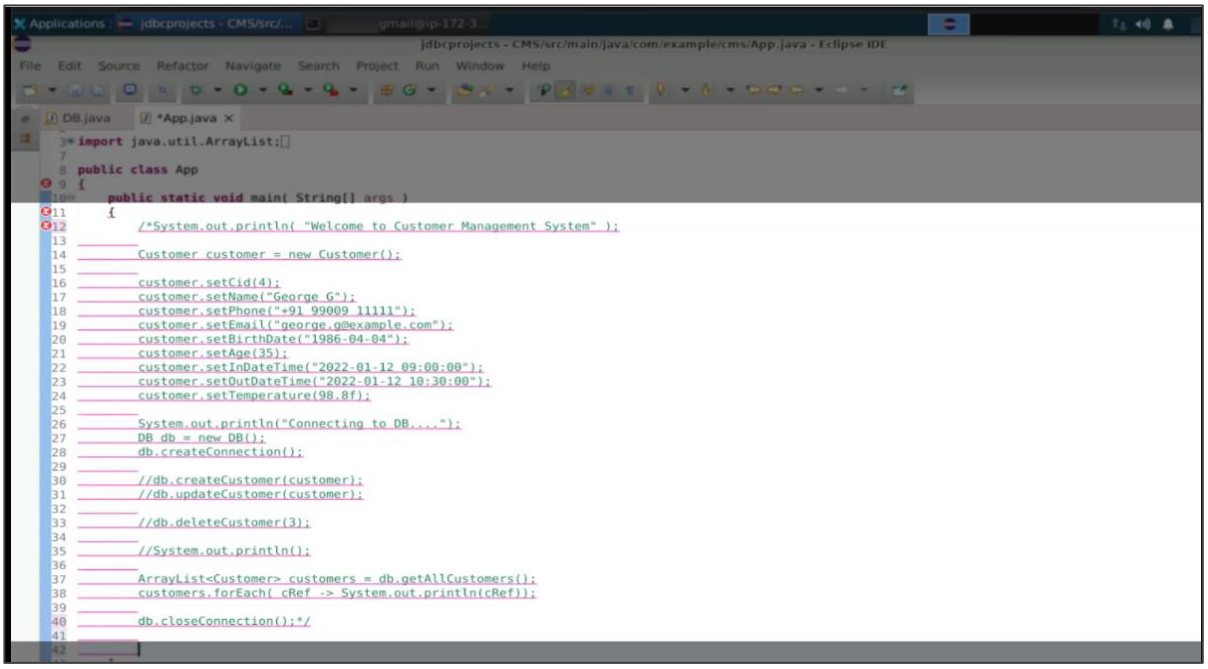
3.3 Call the API known as **callableStatement** from the SQL package for the stored procedure. Use the **callableStatement()** method



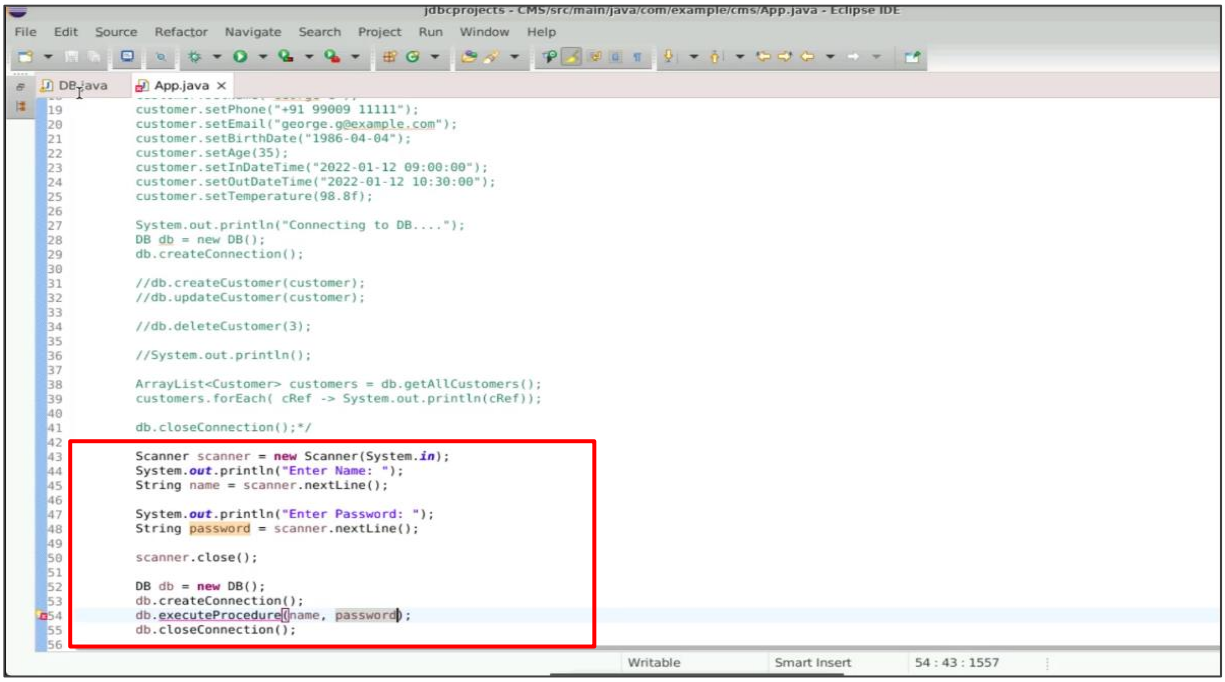
3.4 Initialize the reference variable, create an SQL query, and give substitution to the callableStatement



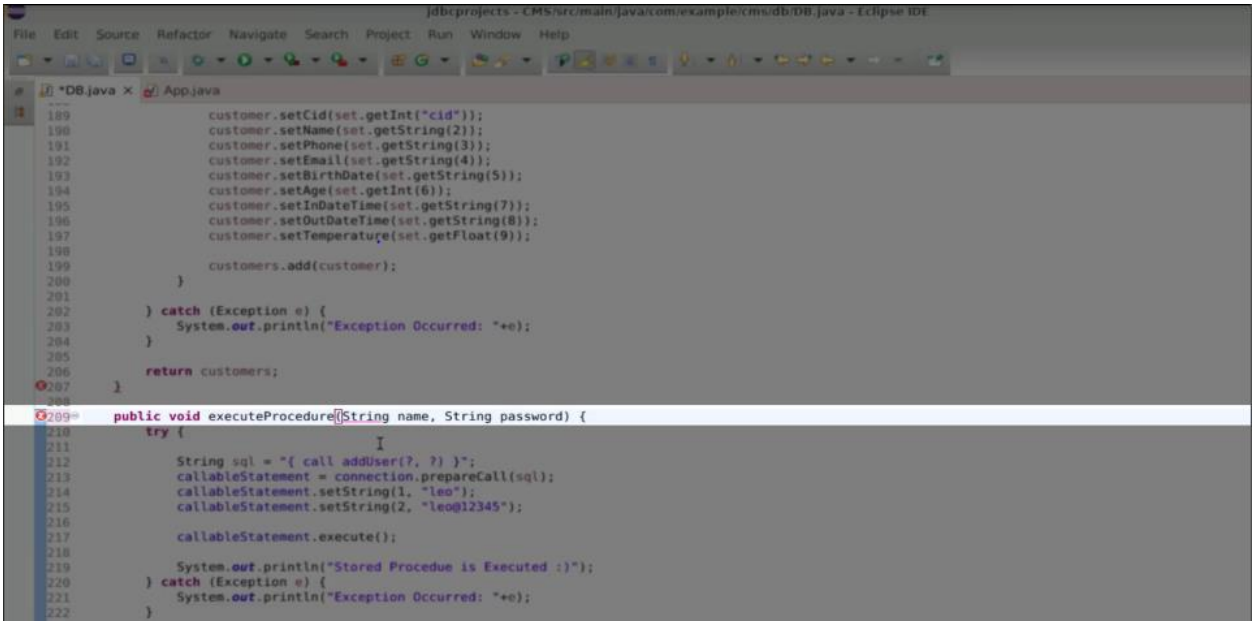
3.5 Go to the **App.java** file and comment on all the existing code available in the **main** method



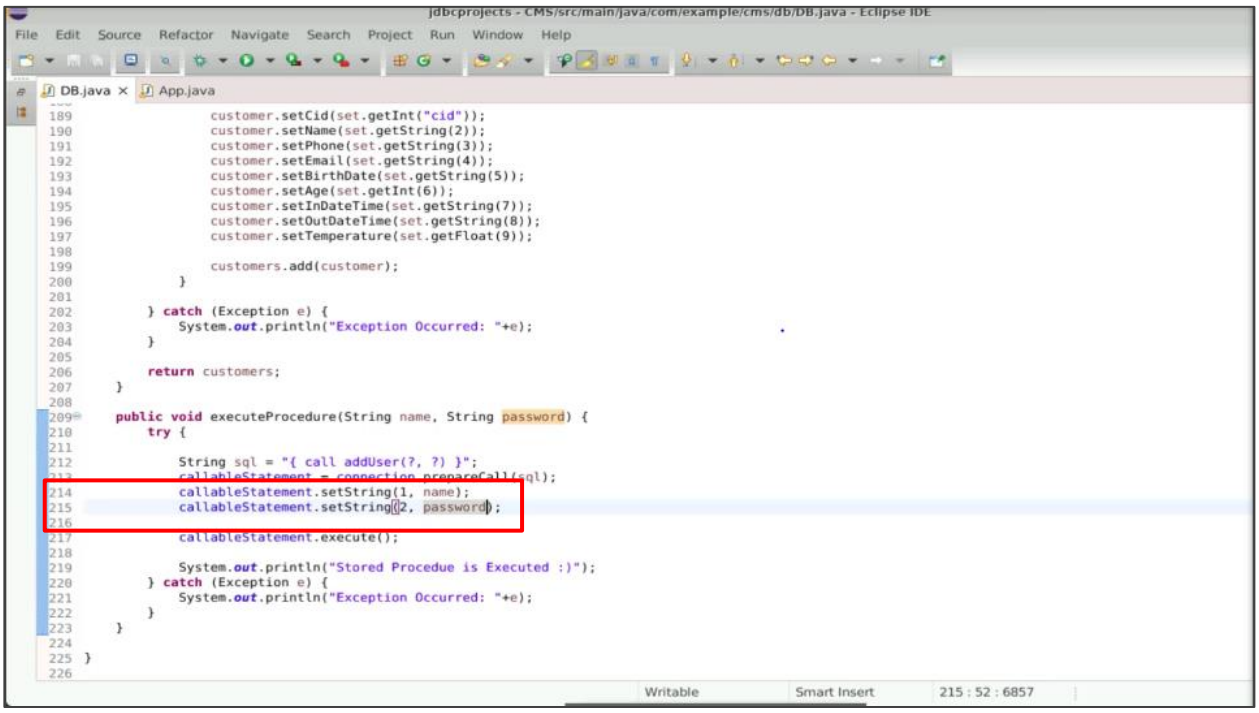
3.6 Create a **scanner** to process the data from the command prompt and create a **db connection** to execute the procedure



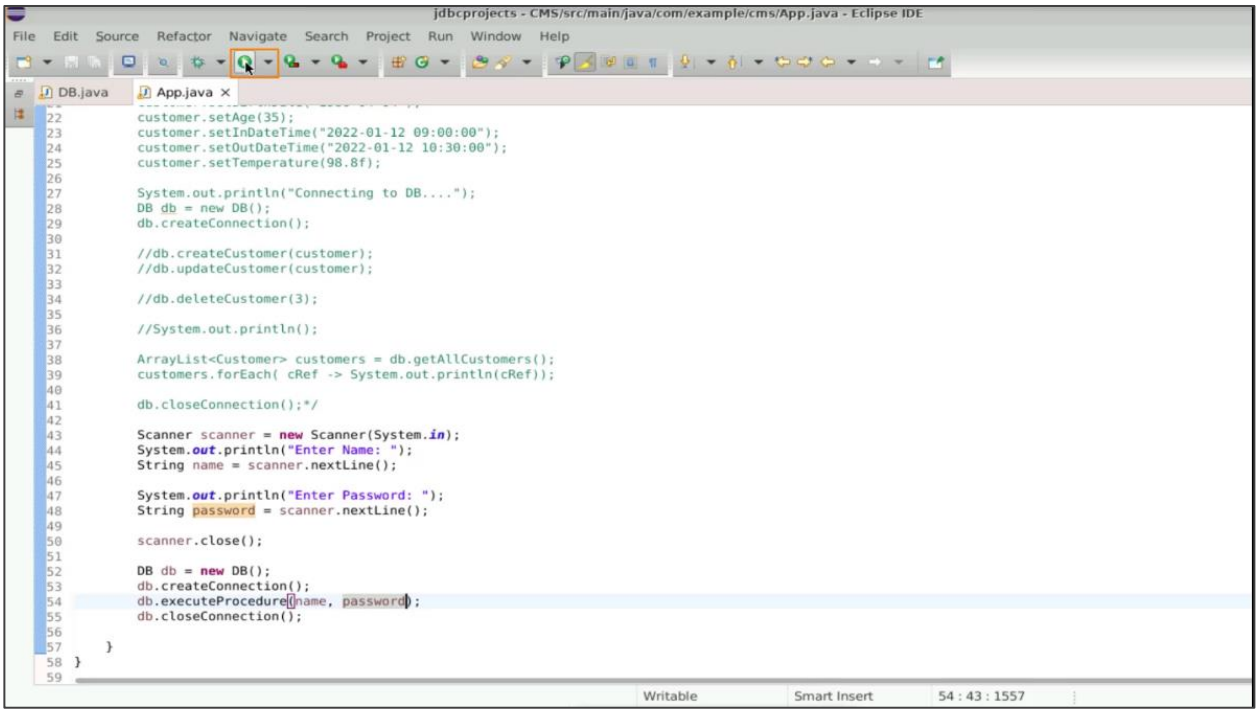
3.7 Go to the **DB.java** file and modify the code to get the name and password as input from the user



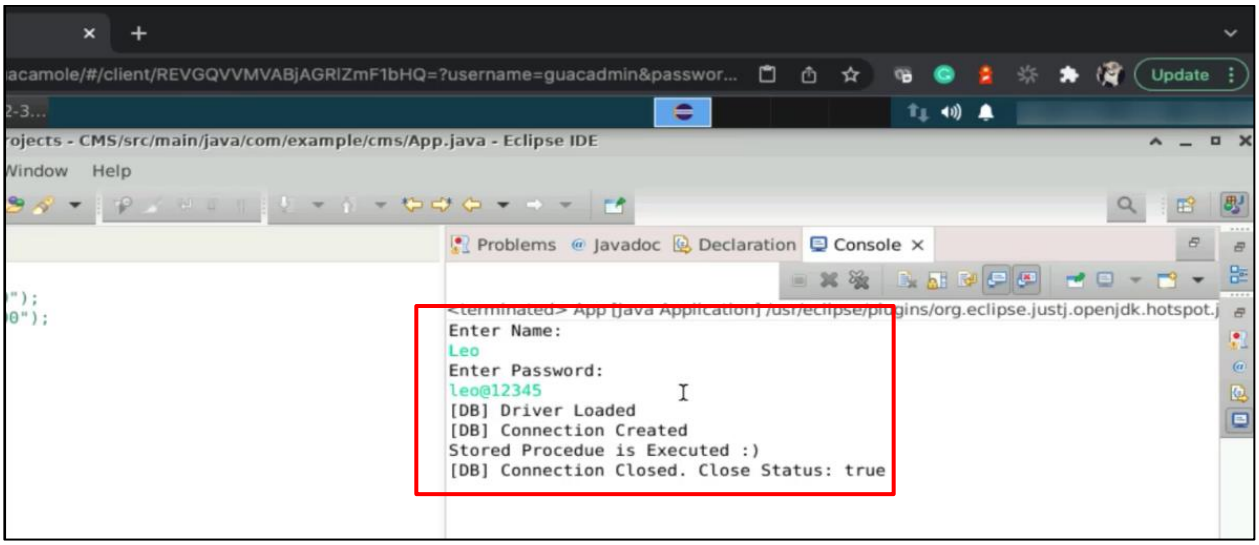
3.8 Modify the code to make the name and password more dynamic



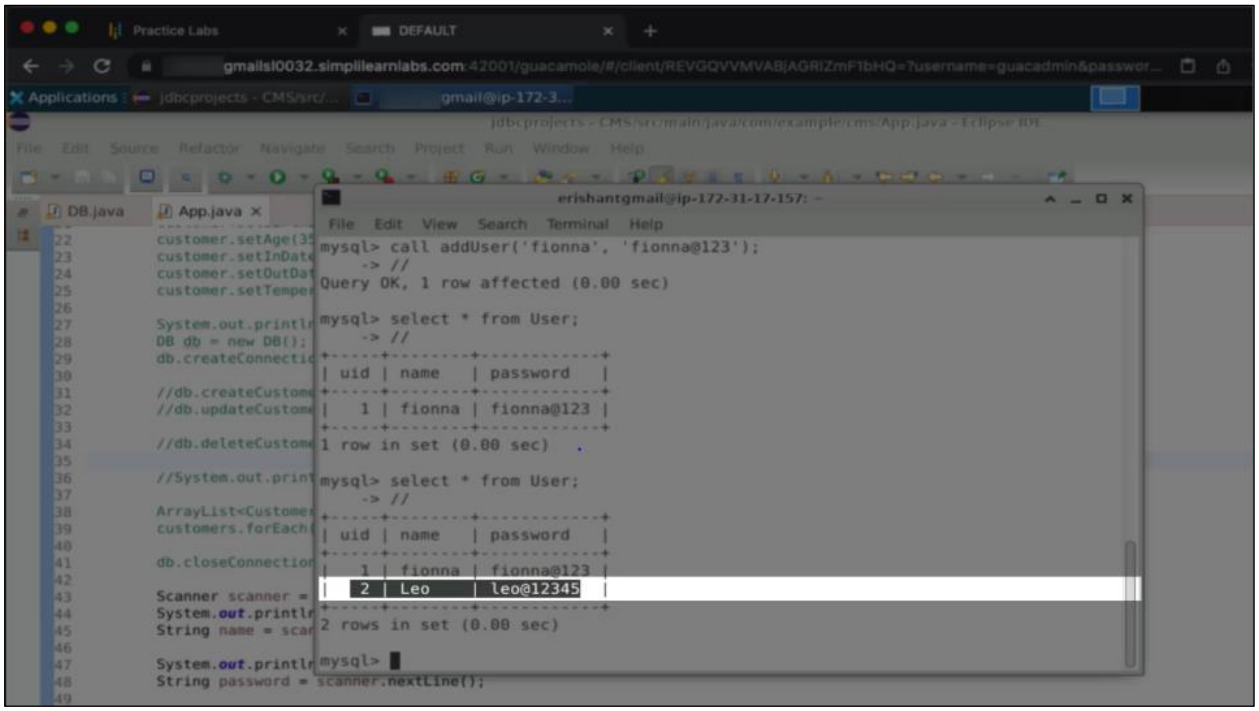
3.9 Go to the **App.java** file and run the program



3.10 Fill in the information and press the **Enter** key in the console. In the output, you can see **Stored Procedure is Executed :)**

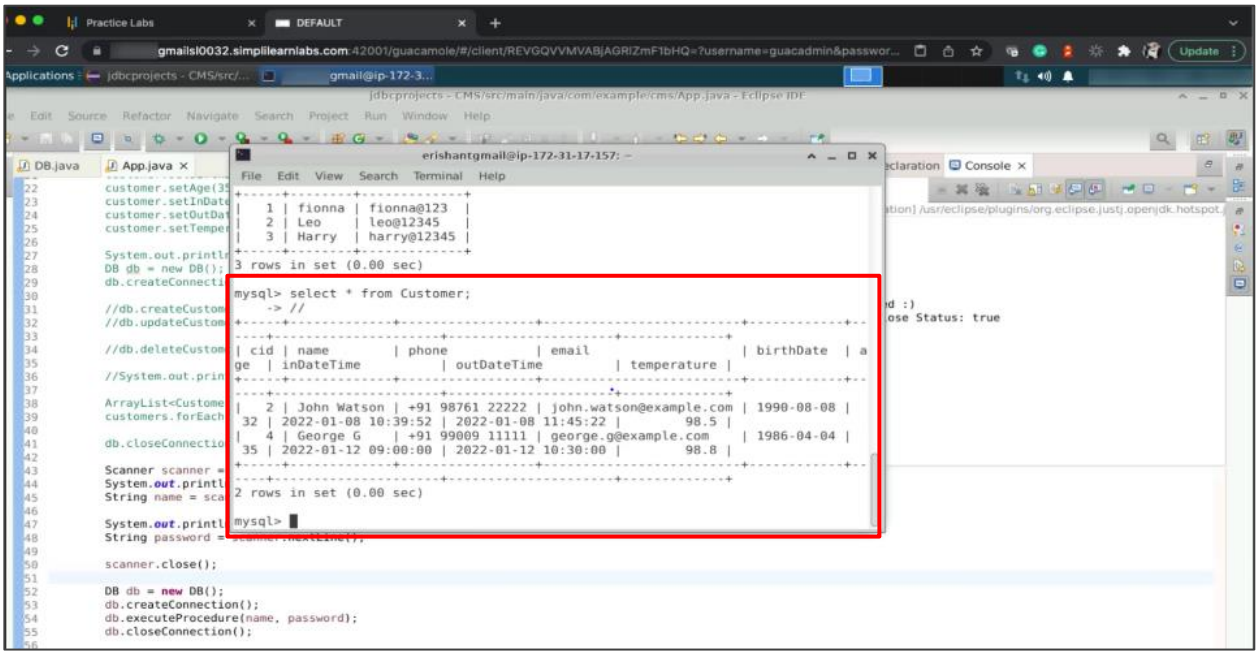


3.11 Go back to the terminal and run the select command again. You can see the user record inserted in the above step in the data table.

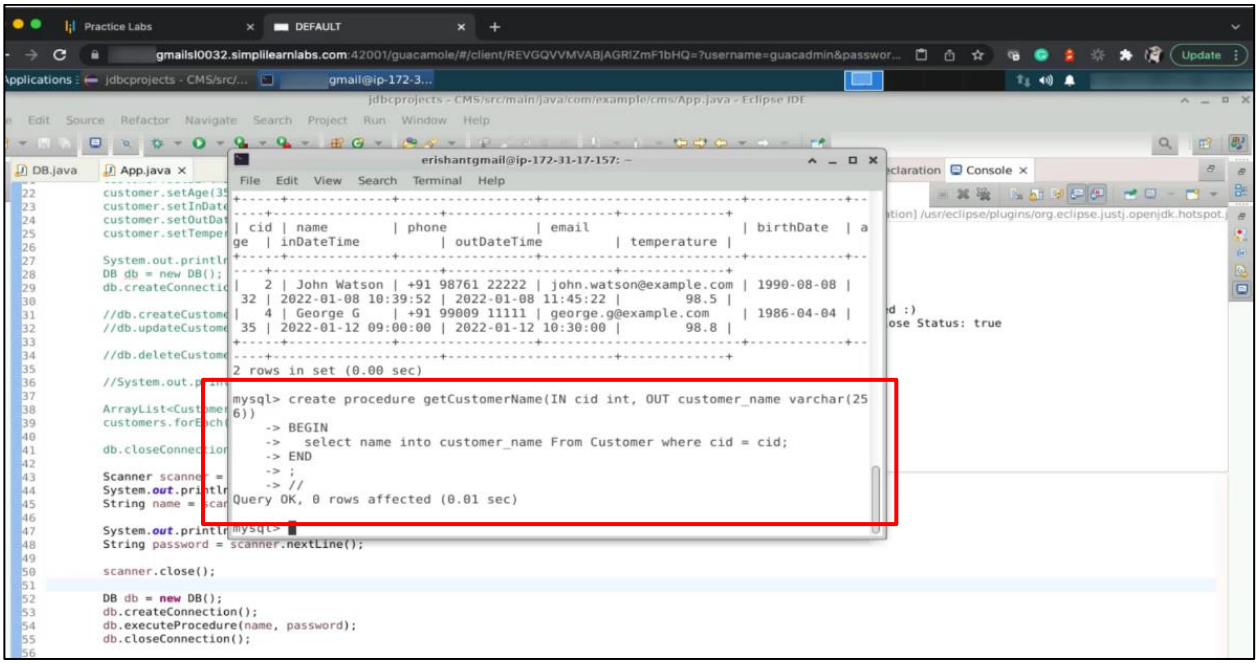


Step 4: Create a stored procedure to read data

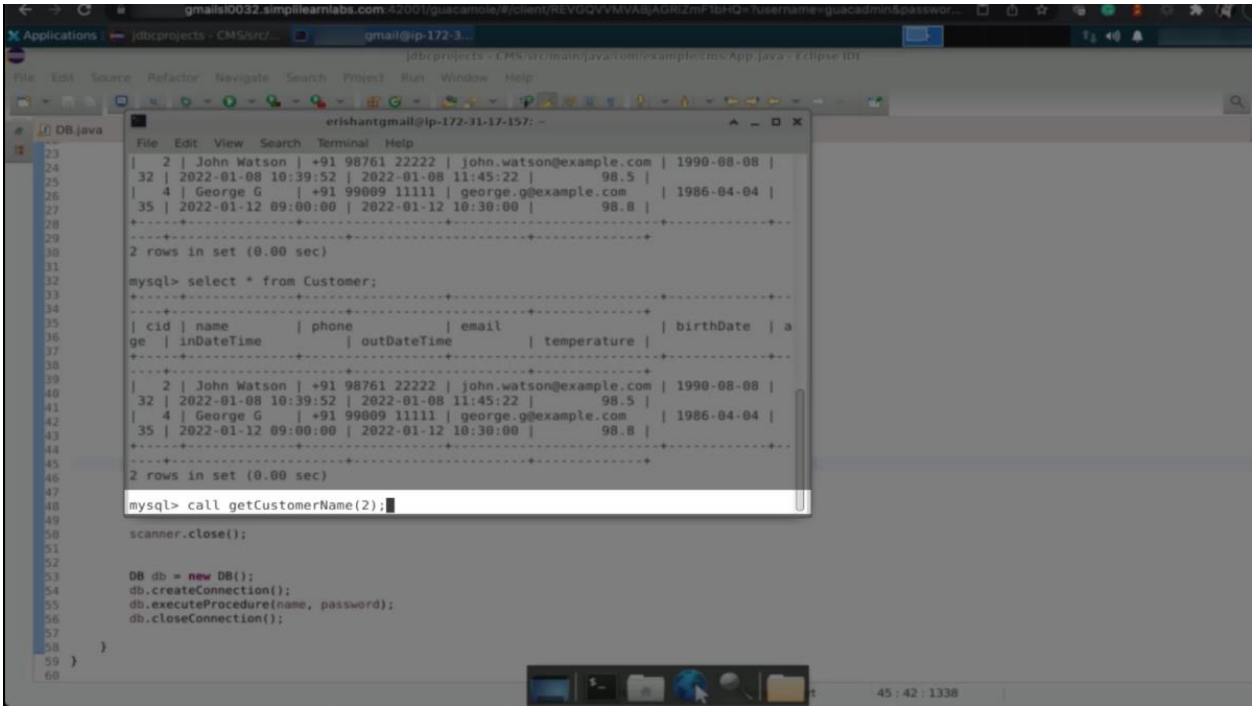
4.1 Run **select * from Customer** to get all customer data



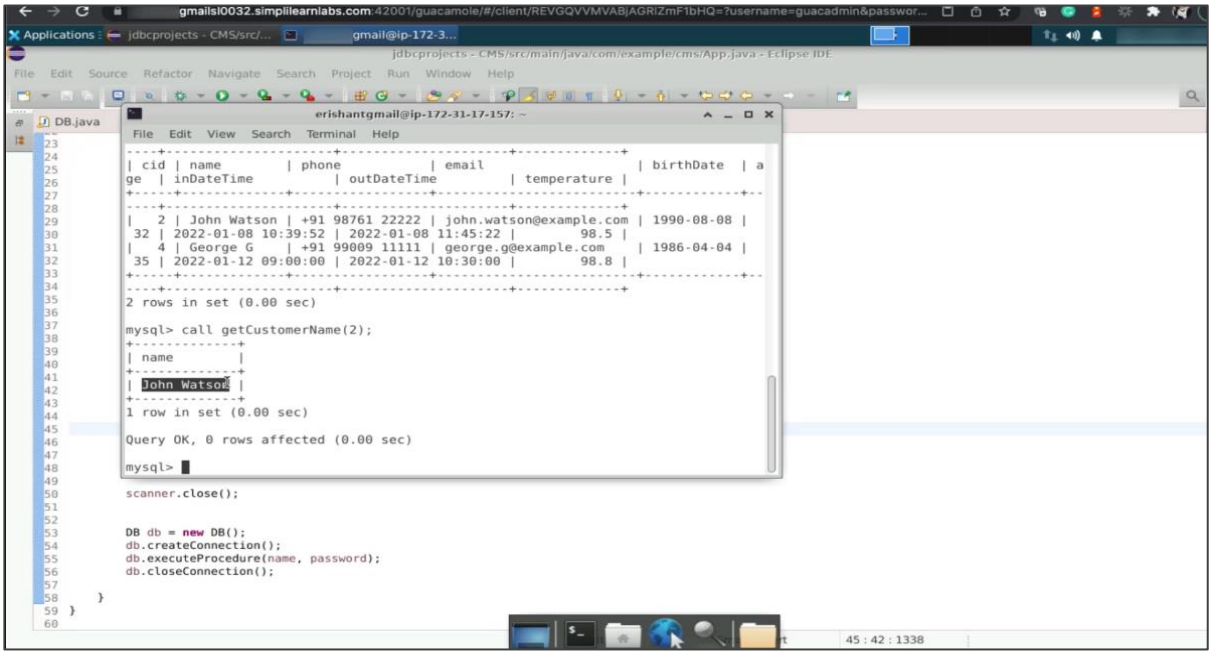
4.2 Create a **stored procedure** as created earlier



4.3 Execute the created stored procedure, run the command **call getCustomerName(2);** where **2** is the customer ID we are calling

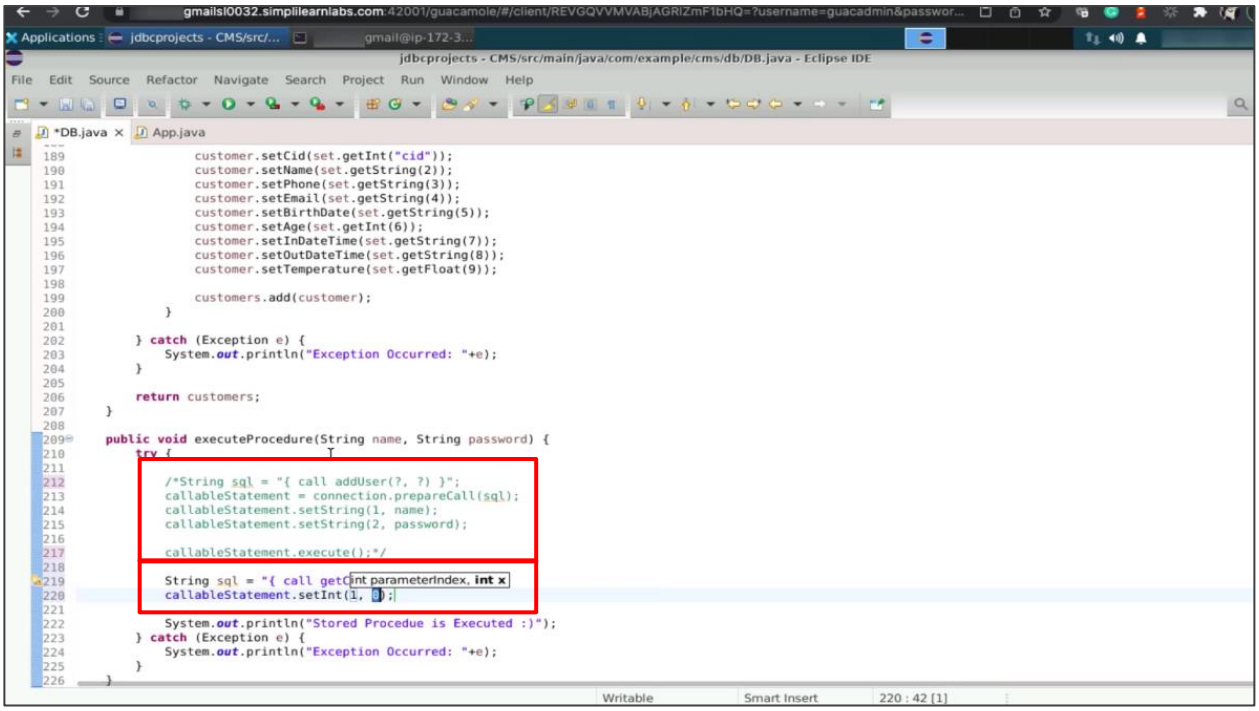


You can see the result with the name of the customer with **cid 2.**

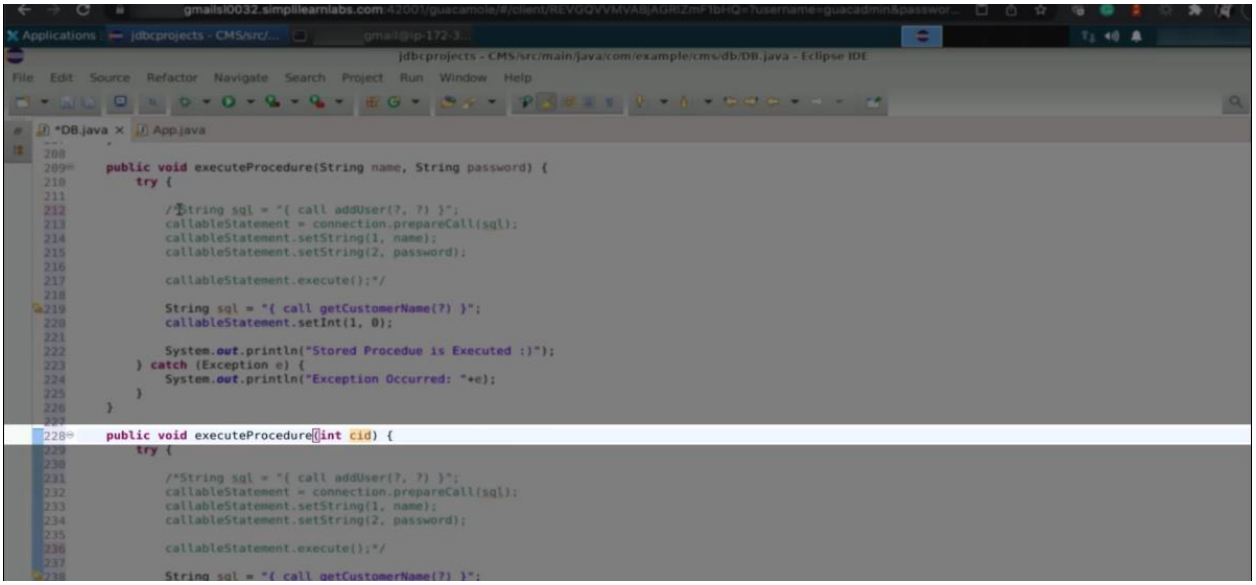


Step 5: Create a method to call a stored procedure to read data

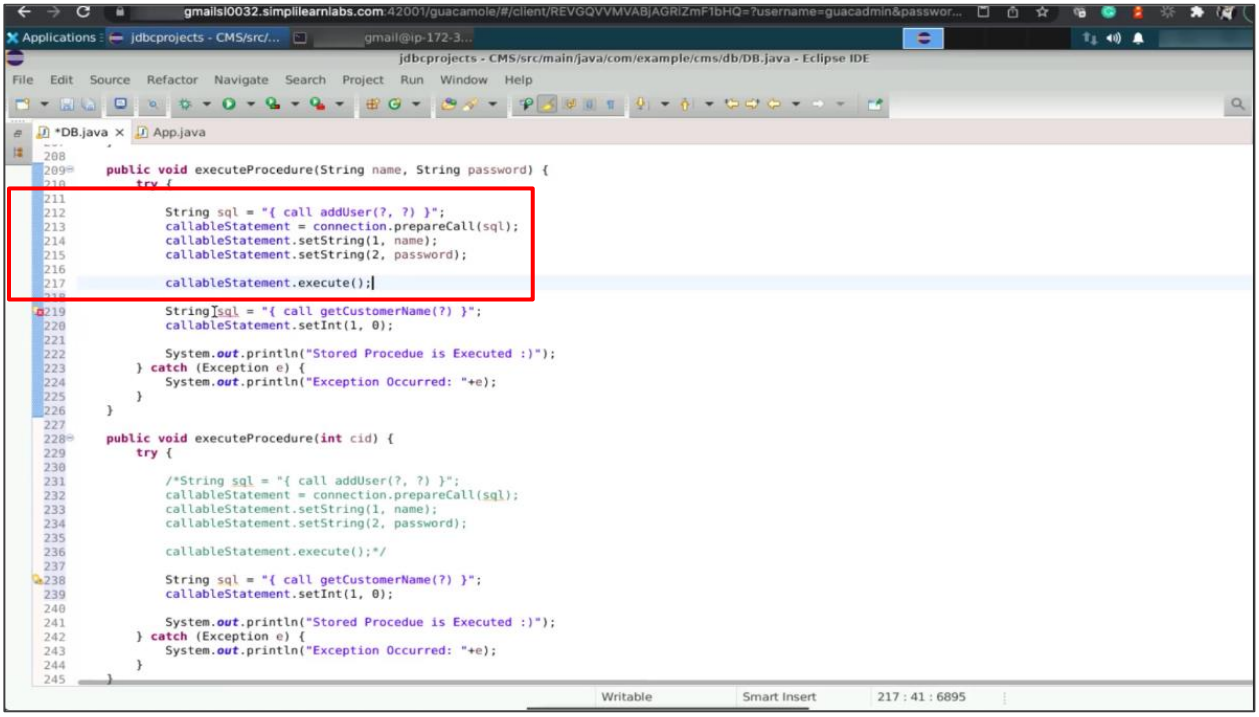
5.1 Open the **DB.java** file and comment out the **try block** in **Eclipse**. Create the SQL statement by passing a single input and using the **prepareCall ()** method



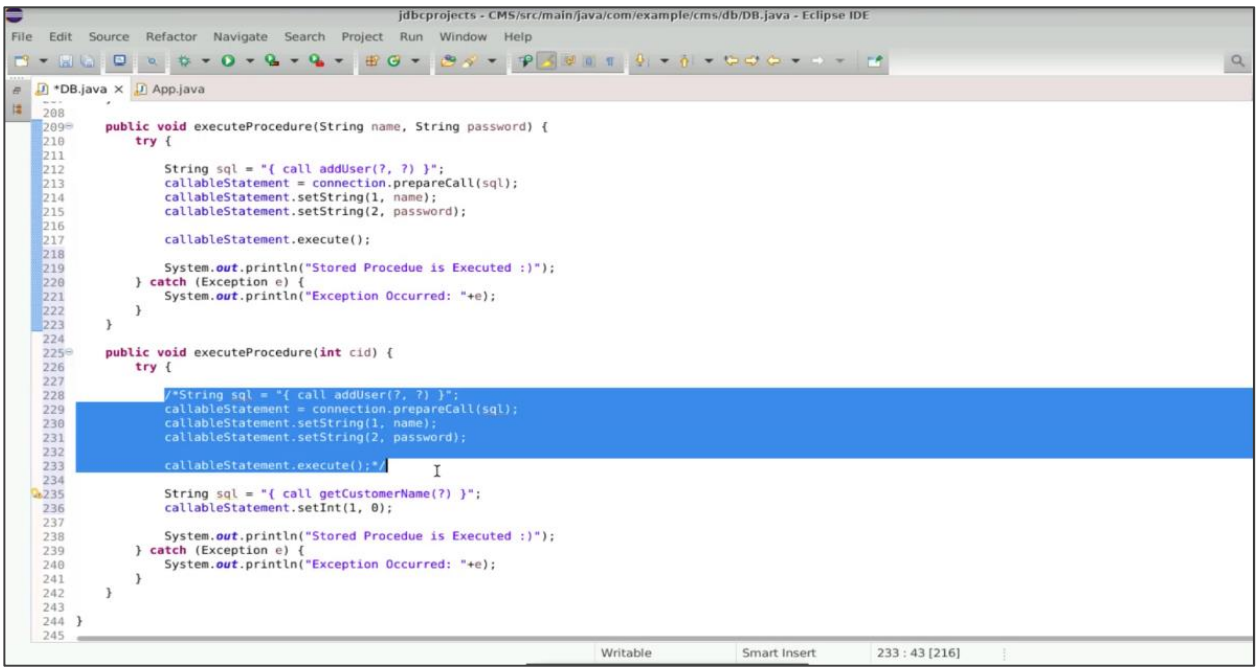
5.2 Create an overloaded version of the method by adding **cid** as an input



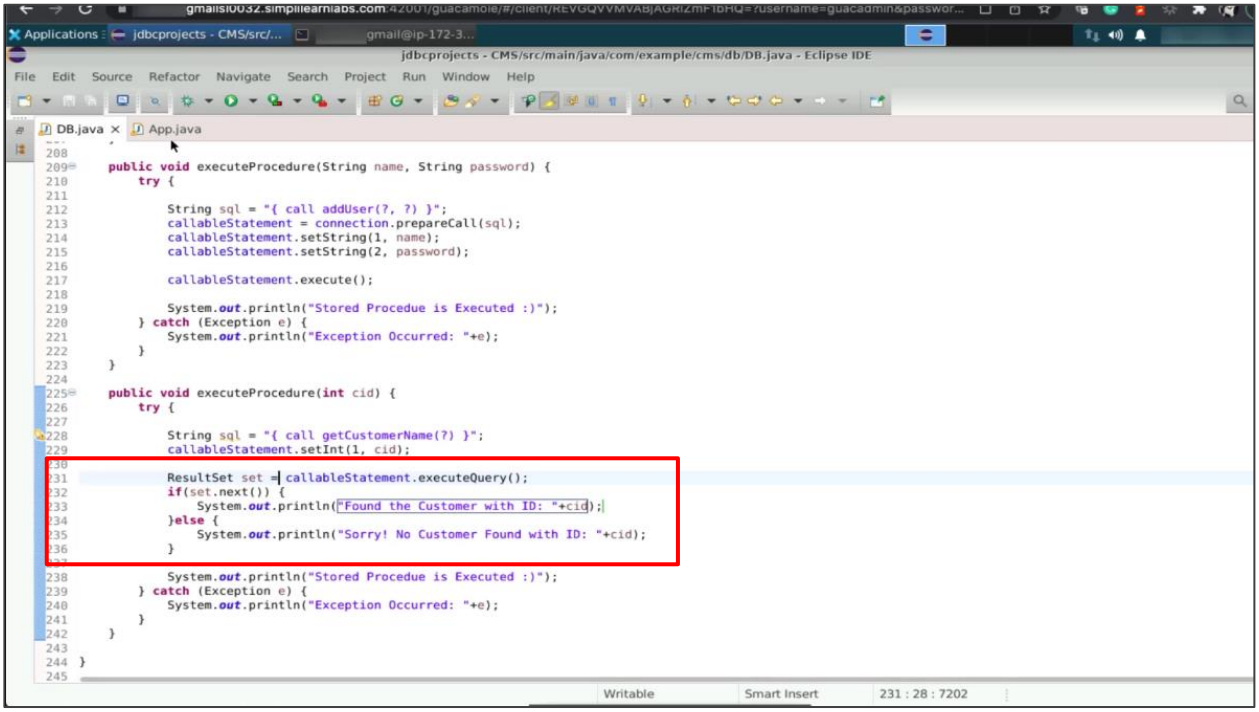
5.3 Uncomment the **try block** of the previous procedure



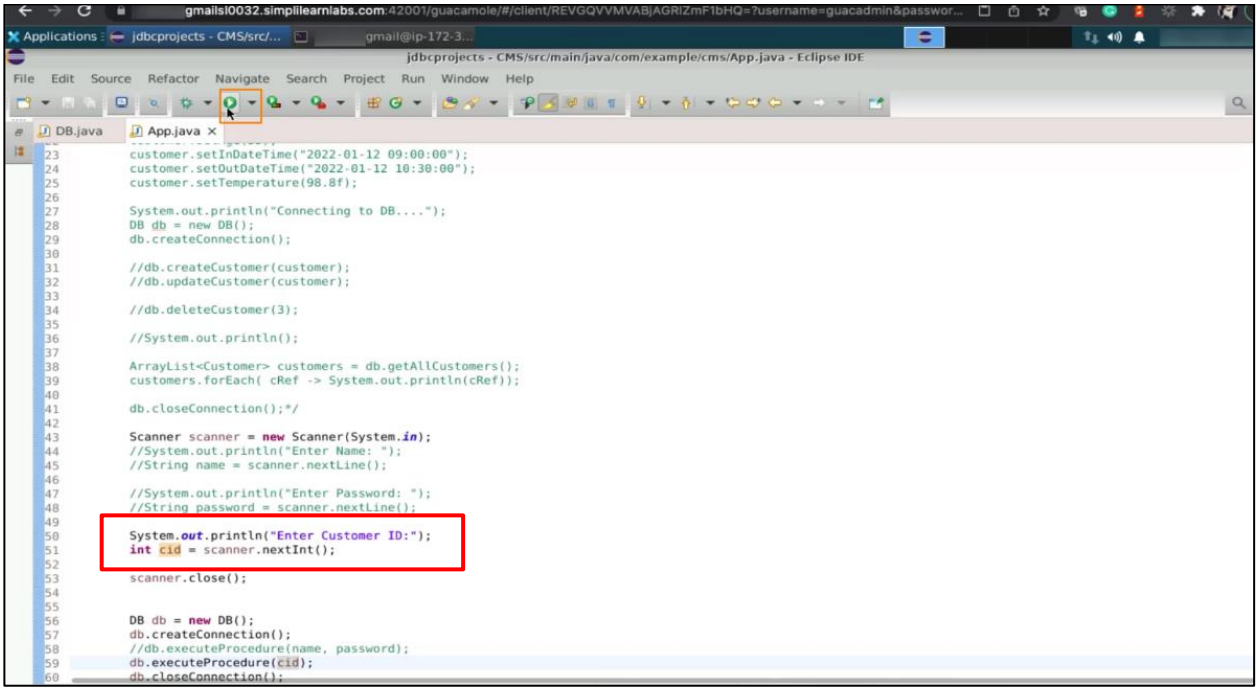
5.4 Delete the **String SQL** and **callableStatement** variables and remove the commented lines from the try block



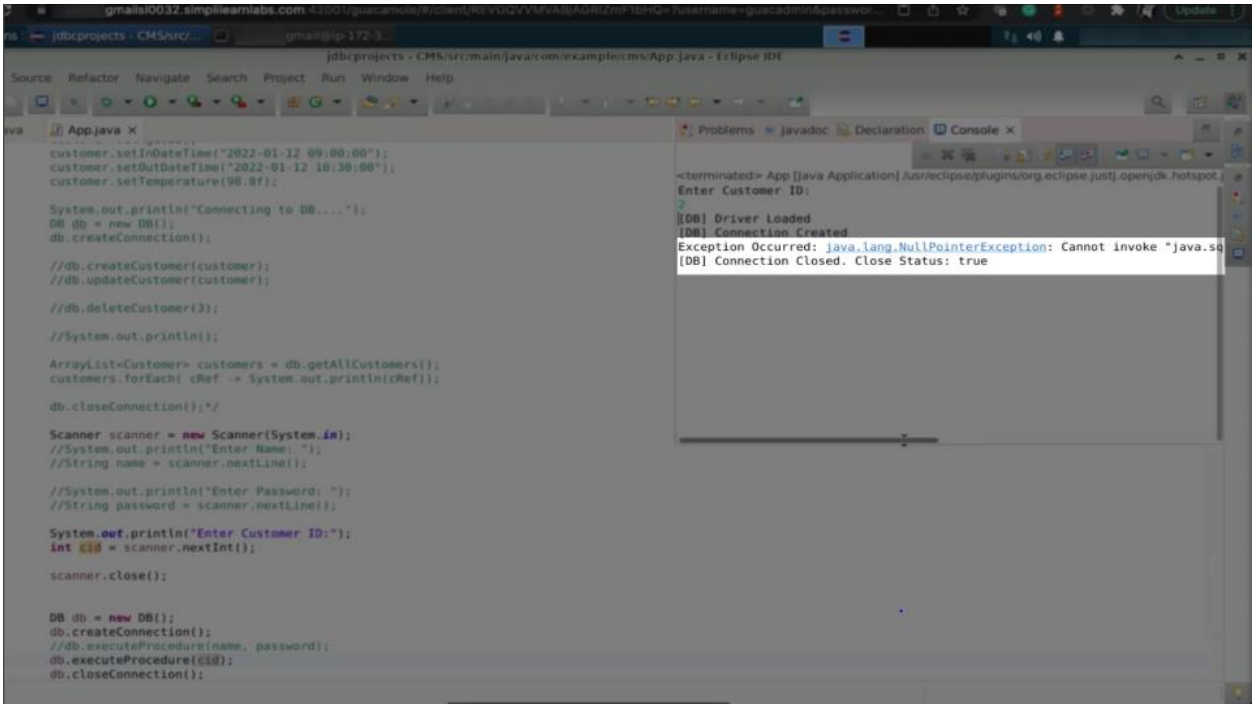
5.5 Create an if-else statement to print the results in all conditions



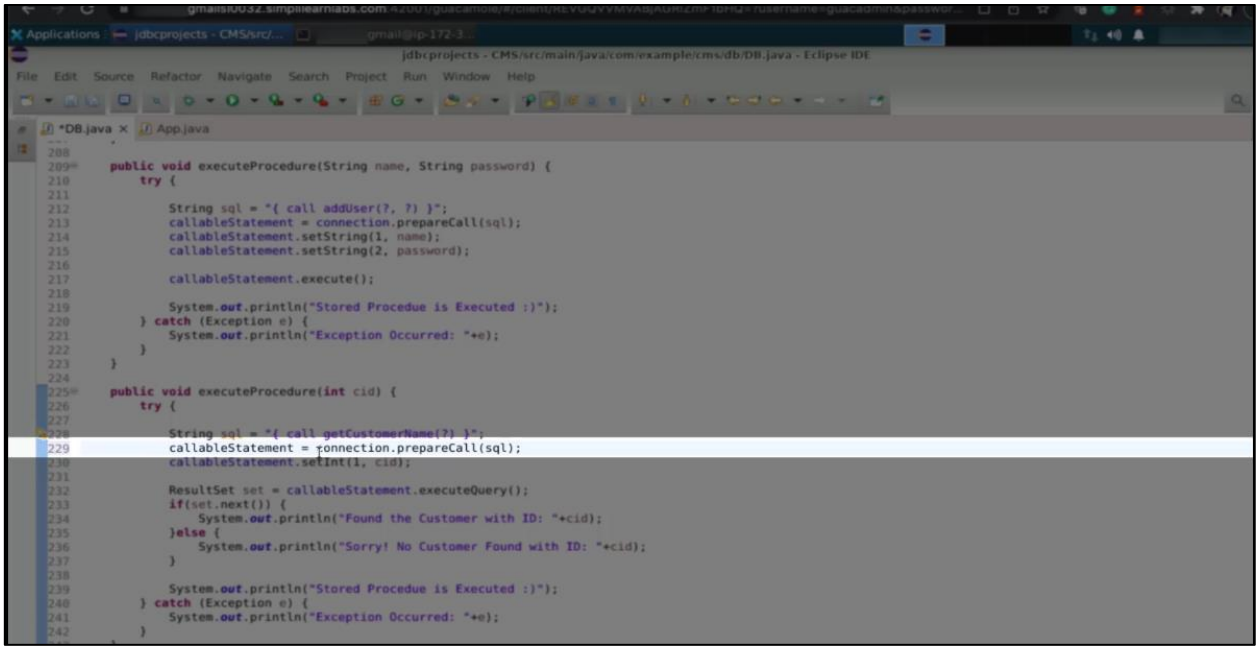
5.6 Add a comment line asking the user to enter the customer ID and run the code



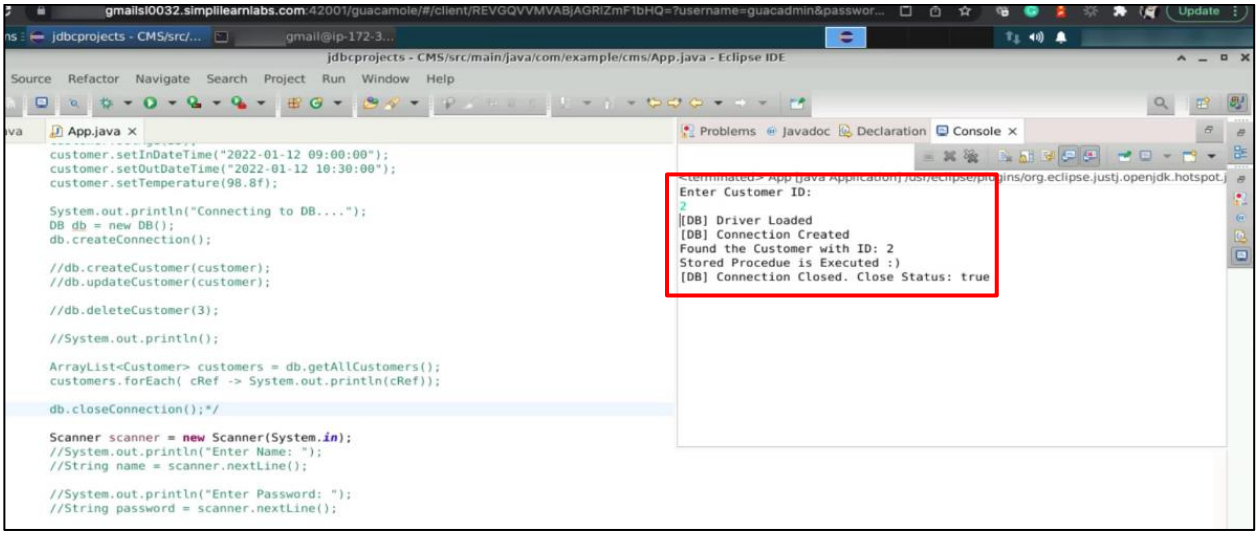
The output with the error **Exception Occurred** can be seen.



5.7 Go to the **DB.java** file and initialize the **callableStatement**. Open the **App.java** file and run the program.



The output **Stored Procedure is Executed :)** is seen.



By following these steps, you have successfully created stored procedures and executed them using the JDBC API called **callableStatement** for efficient and reusable database operations. This approach enables the direct invocation of stored procedures defined in the database from a Java application.