Lesson 01 Demo 03
Performing CRUD Operations

Objective: To perform create, read, update, and delete operations on the created database for managing customer records effectively

Tool required: Eclipse IDE

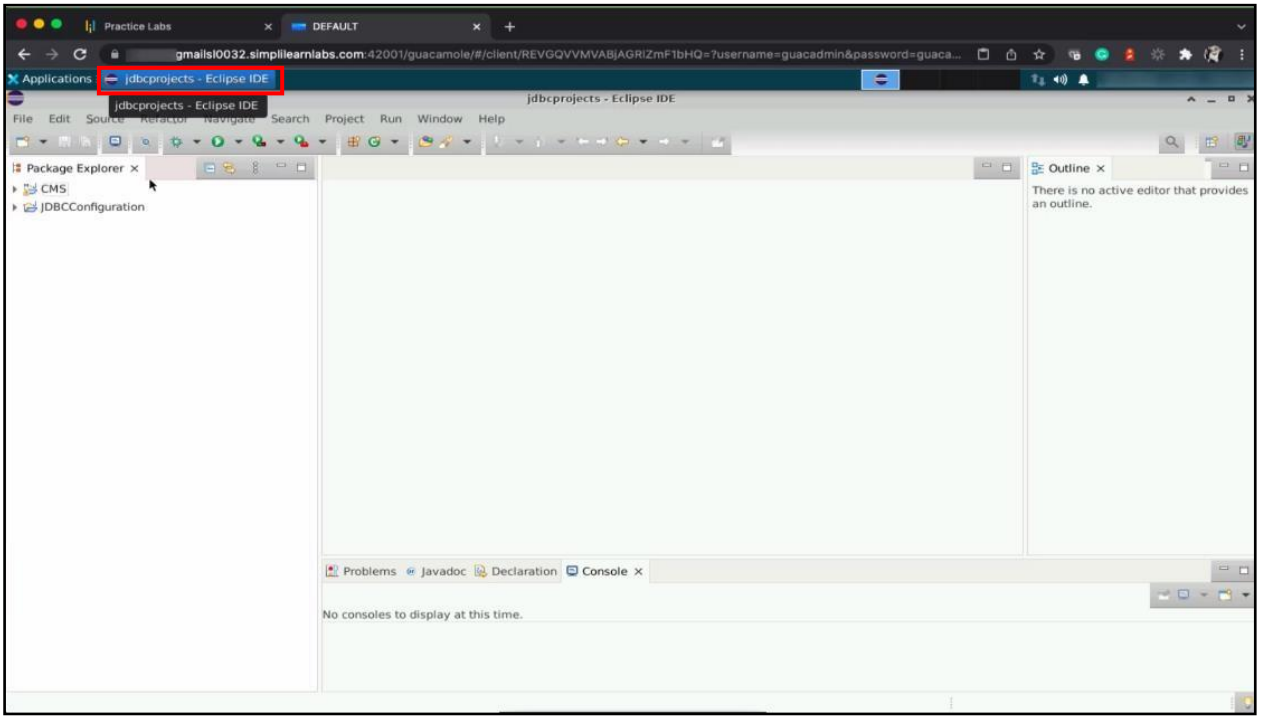
Prerequisites: Lesson 01 Demo 02

Steps to be followed:

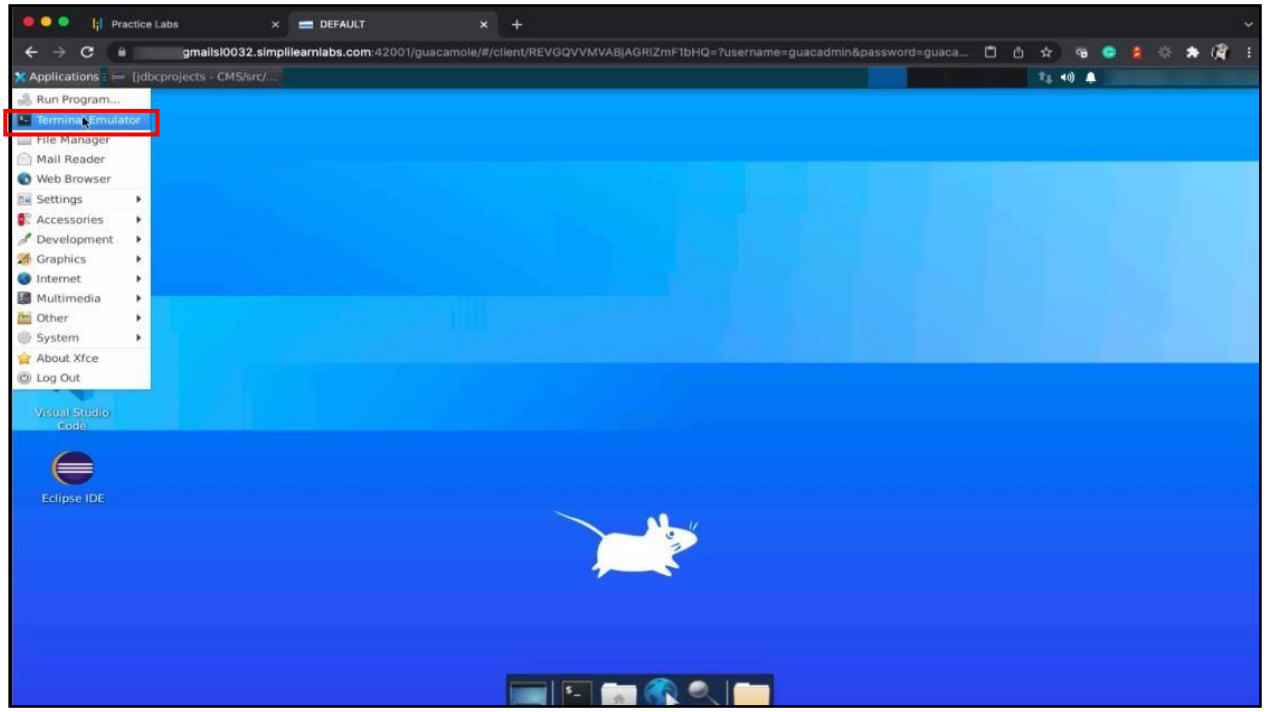
1. Open the Terminal Emulator
2. Write an insert operation in Eclipse IDE
3. Create an SQL query for the insert operation
4. Write an update operation
5. Perform the getallcustomer operations
6. Perform the delete operation

Step 1: Open the Terminal Emulator

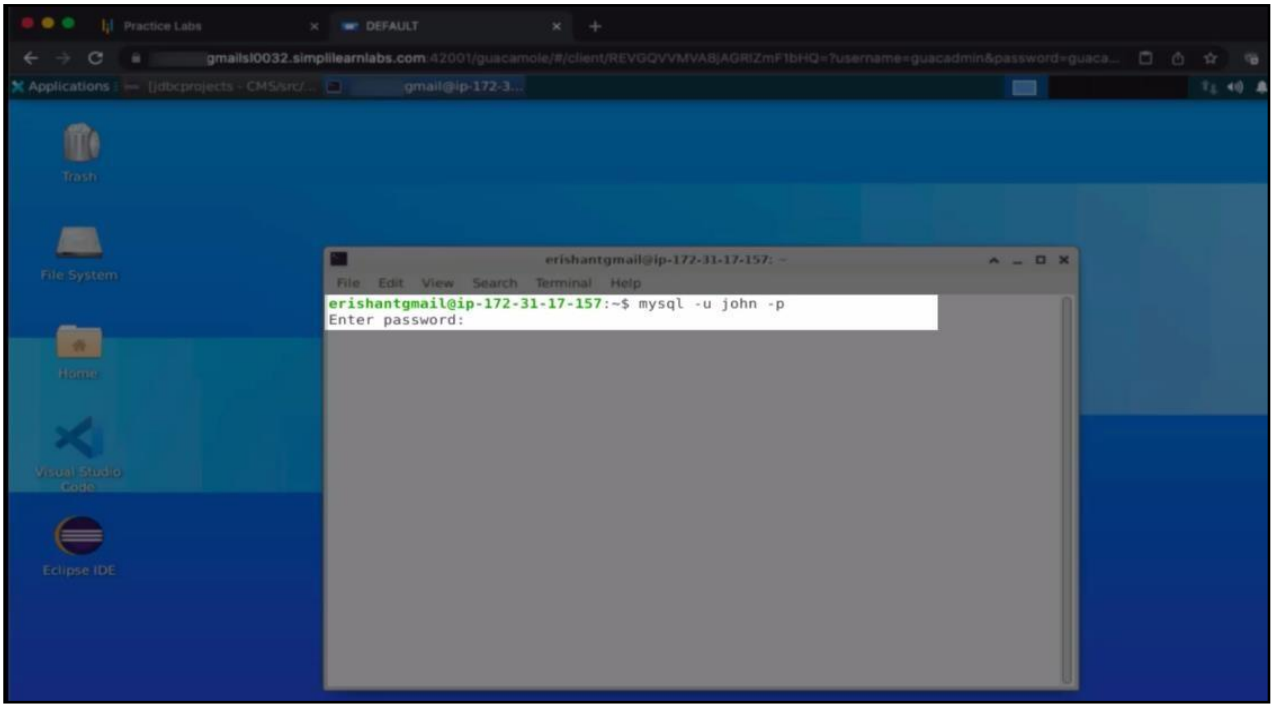
1.1 Open **Eclipse IDE**



1.2 Open the **Terminal Emulator**

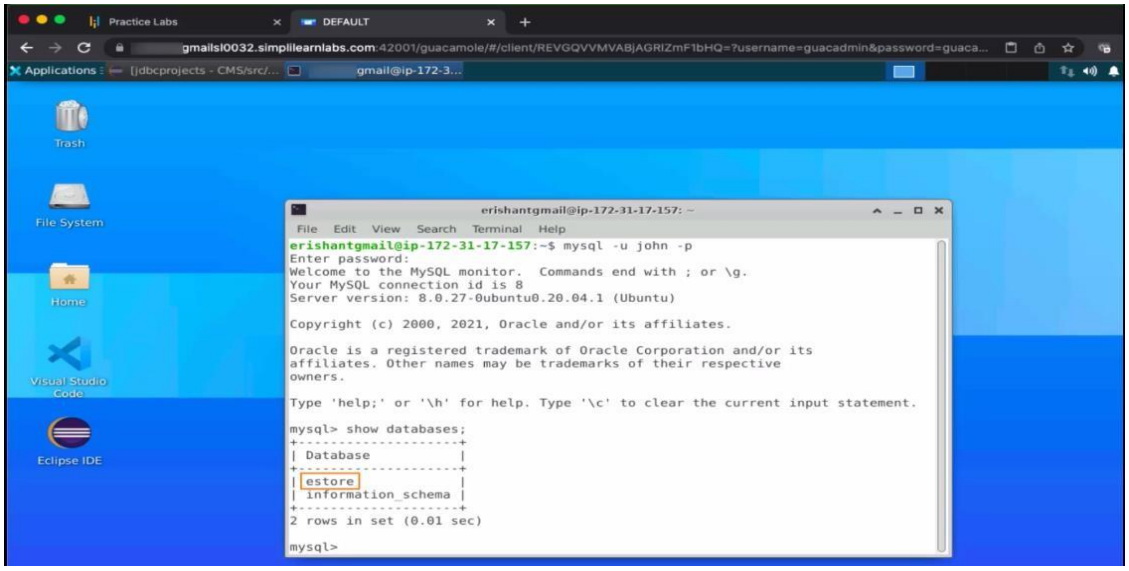


1.3 Log in to MySQL using the command **mysql -u john -p**

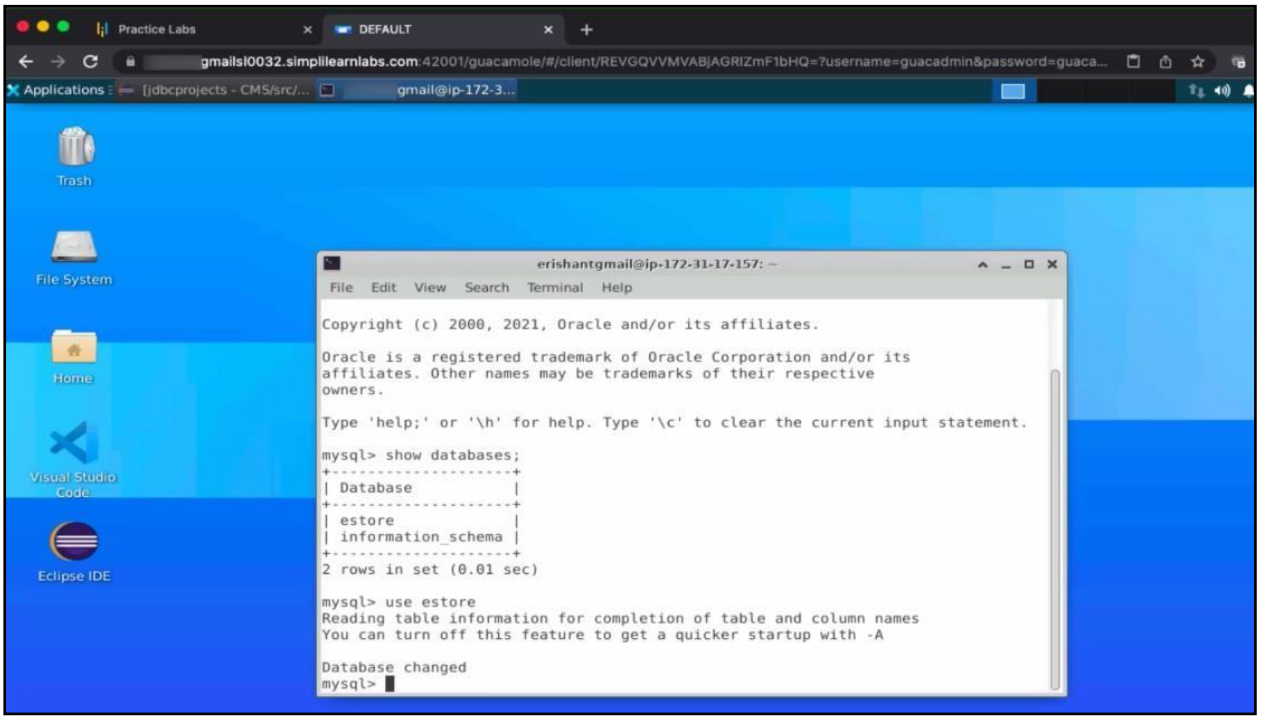


Note: A user named **John** has already created for the database.

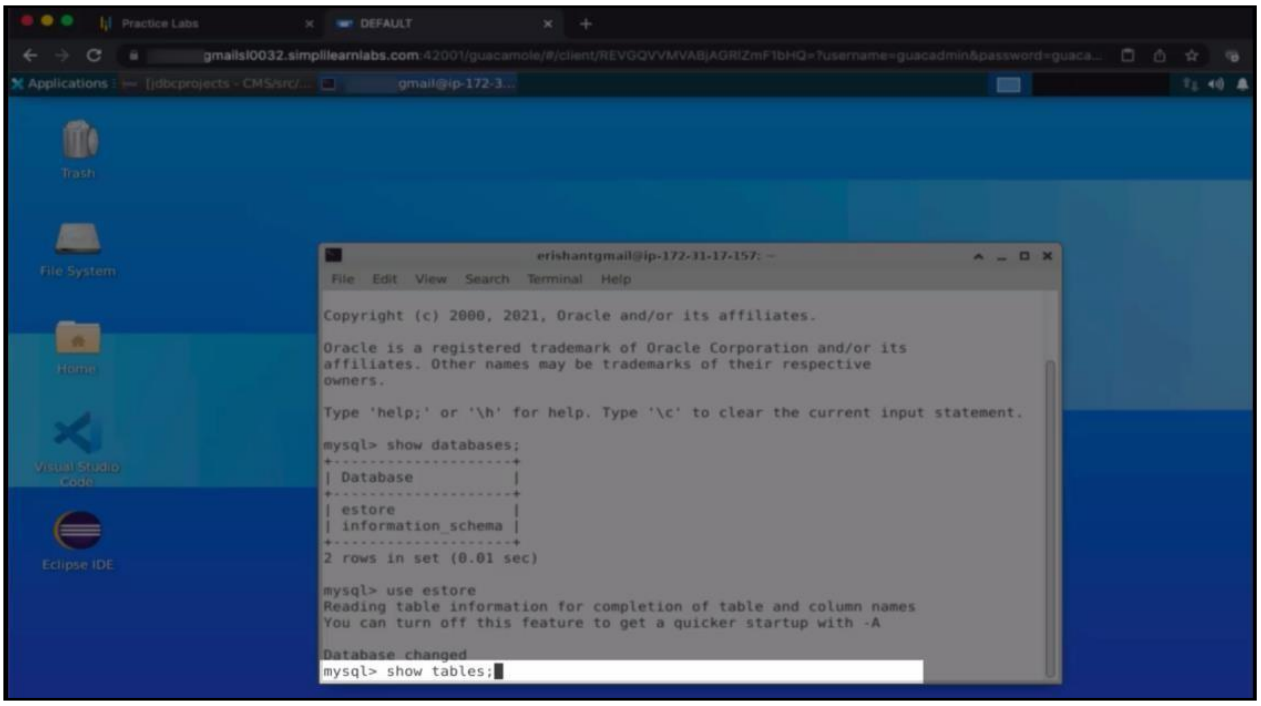
1.4 Type **show databases;** and list all the databases available

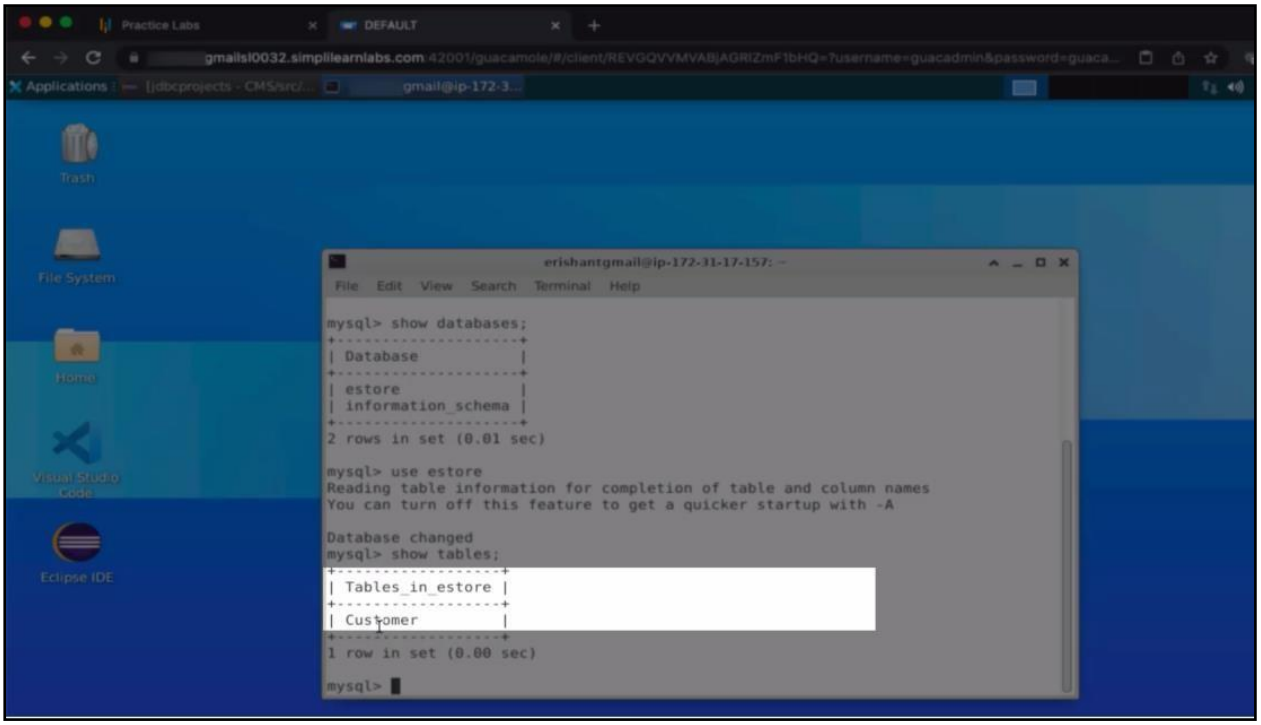


1.5 Type **use estore;** and change the database to **estore** database

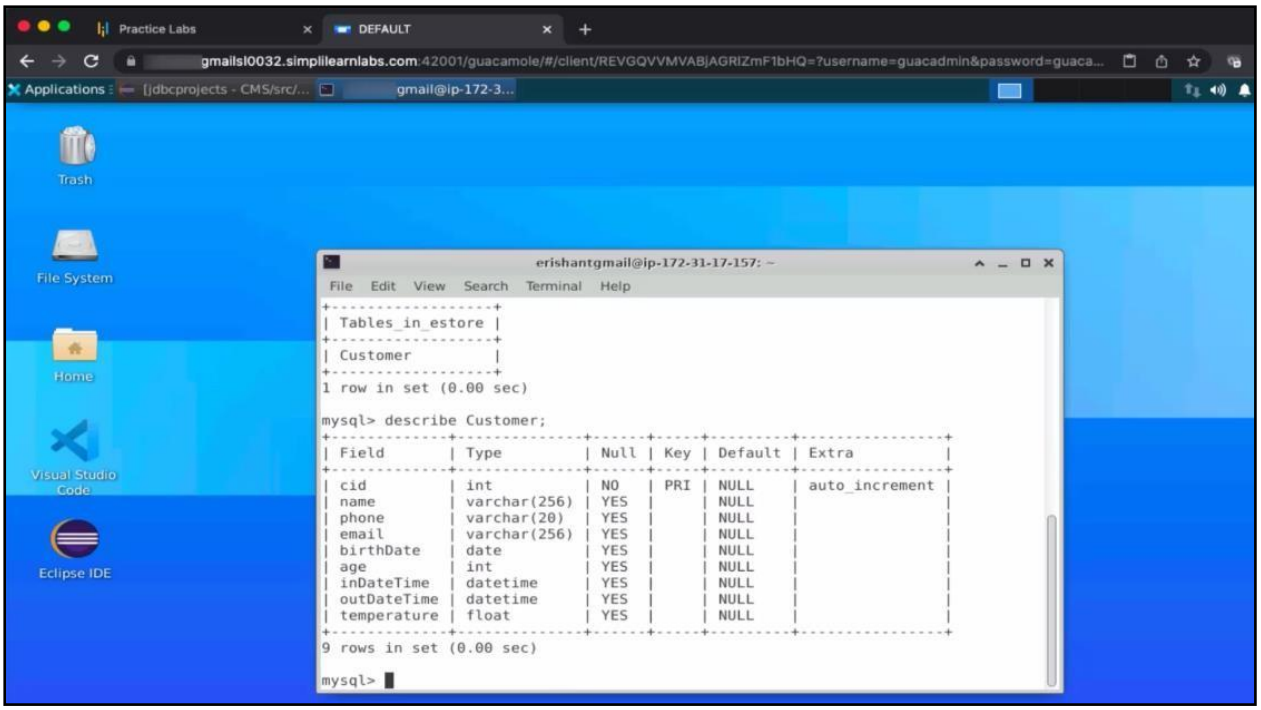


1.6 Type **show tables;** to list the tables available in the **estore** database

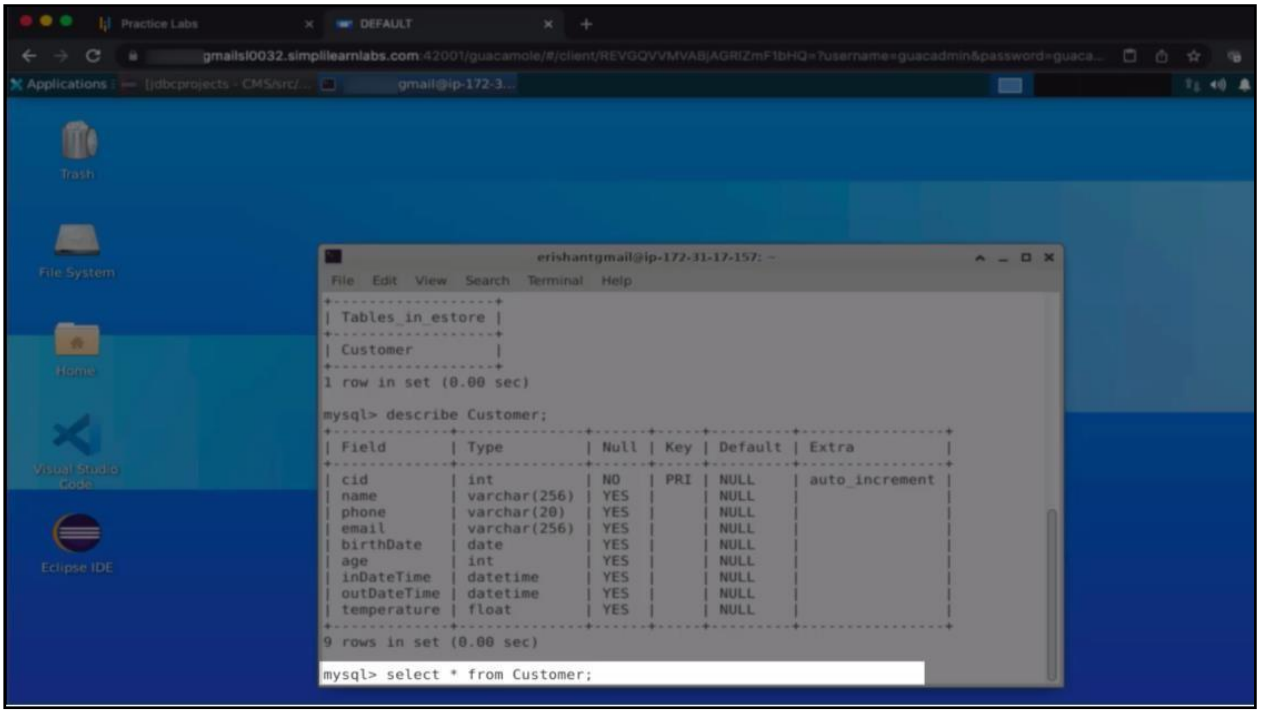


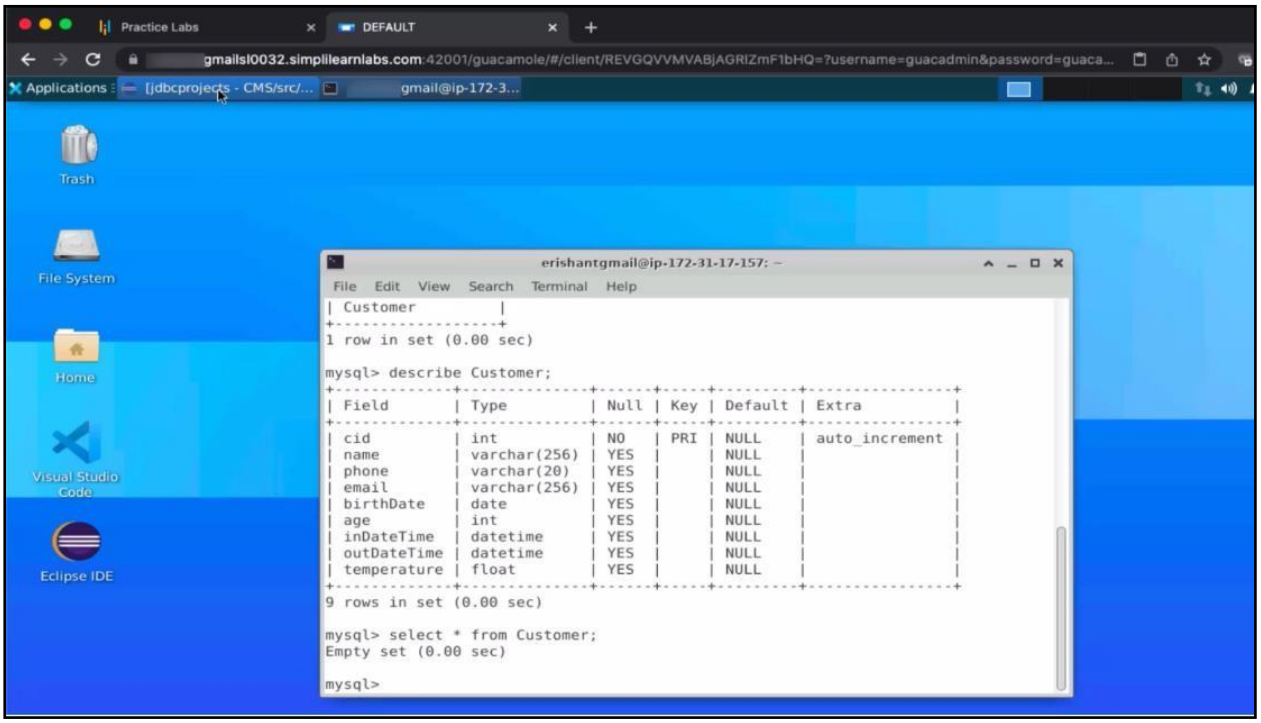


1.7 Type **describe Customer;** to get detailed information on the table



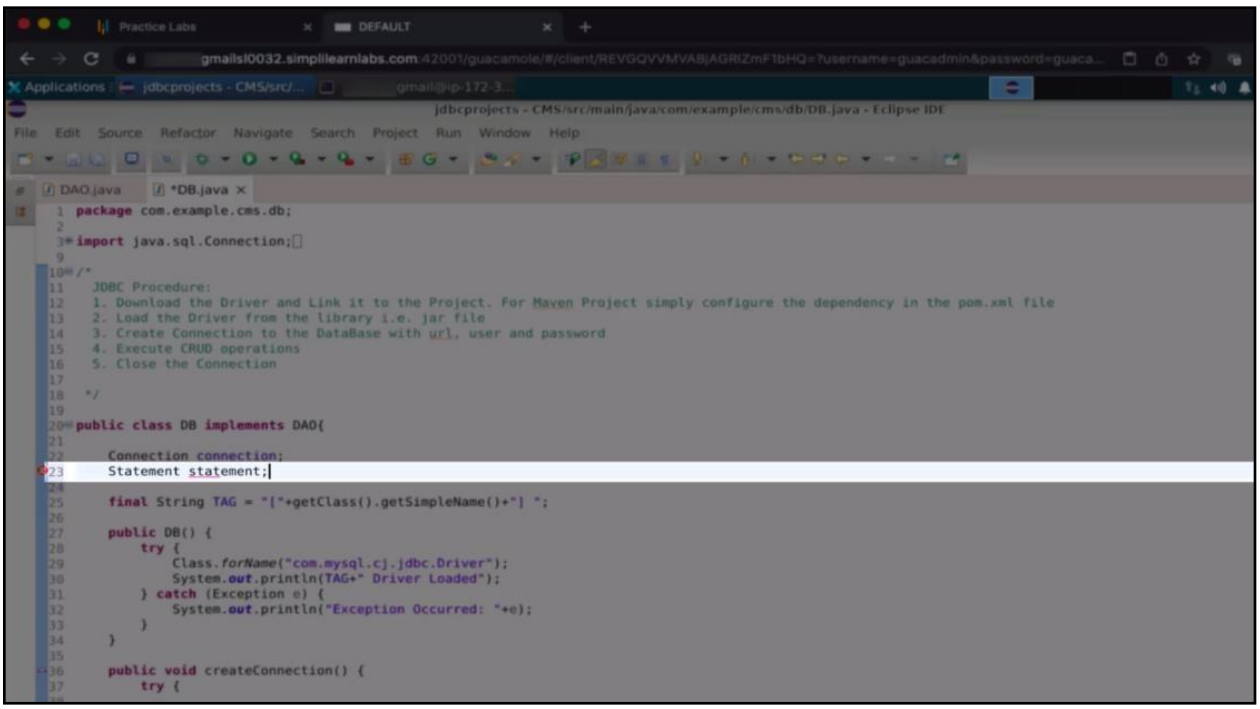
1.8 Type **select * from Customer;** to select data from the table




Practice Labs
DEFAULT
gmailsl0032.simplilearnlabs.com:42001/guacamole/#/client/REVGQVVMVABjAGRIZmF1bHQ=?username=guacadmin&password=guaca...
Applications
[jdbcprojects - CMS/src/...
gmail@ip-172-3...
Trash
File System
Home
Visual Studio Code
Eclipse IDE
erishantgmail@ip-172-31-17-157: ~
File Edit View Search Terminal Help
| Customer |
+----------+
1 row in set (0.00 sec)
mysql> describe Customer;
+-------------+-------------+------+-----+---------+----------------+
| Field | Type | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+----------------+
cid	int	NO	PRI	NULL	auto_increment
name	varchar(256)	YES		NULL	
phone	varchar(20)	YES		NULL	
email	varchar(256)	YES		NULL	
birthDate	date	YES		NULL	
age	int	YES		NULL	
inDateTime	datetime	YES		NULL	
outDateTime	datetime	YES		NULL	
temperature	float	YES		NULL	
+-------------+-------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
mysql> select * from Customer;
Empty set (0.00 sec)
mysql>

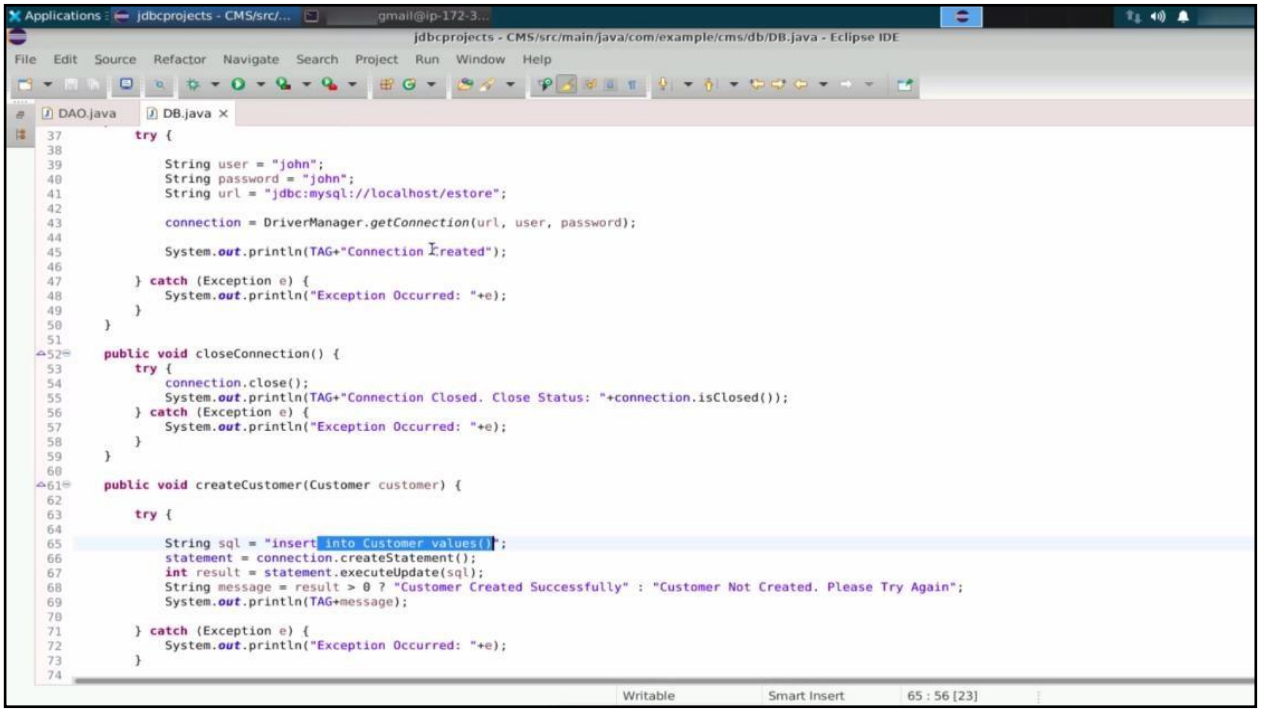
Step 2: Write an insert operation in Eclipse IDE

2.1 Handle exception errors by typing the API call statement using the **Statement** object

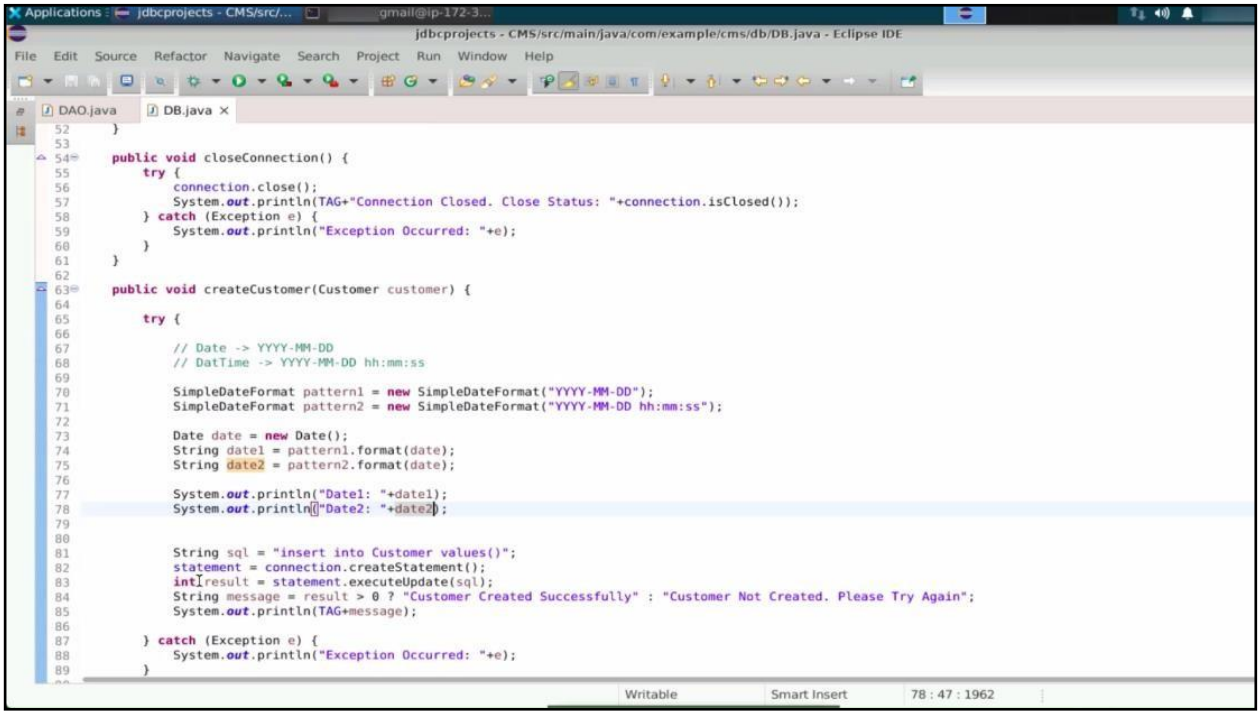


2.2 Type the insert operation inside the **try-catch** block:

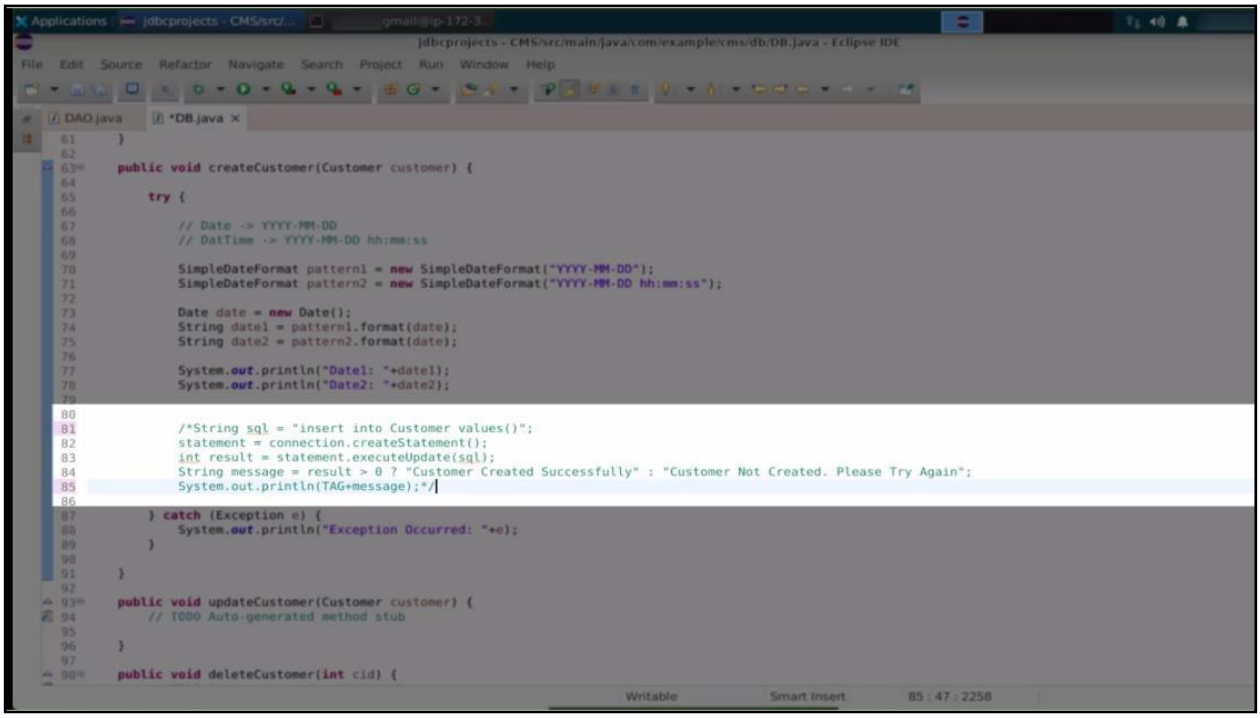
String sql = “insert into Customer values()”



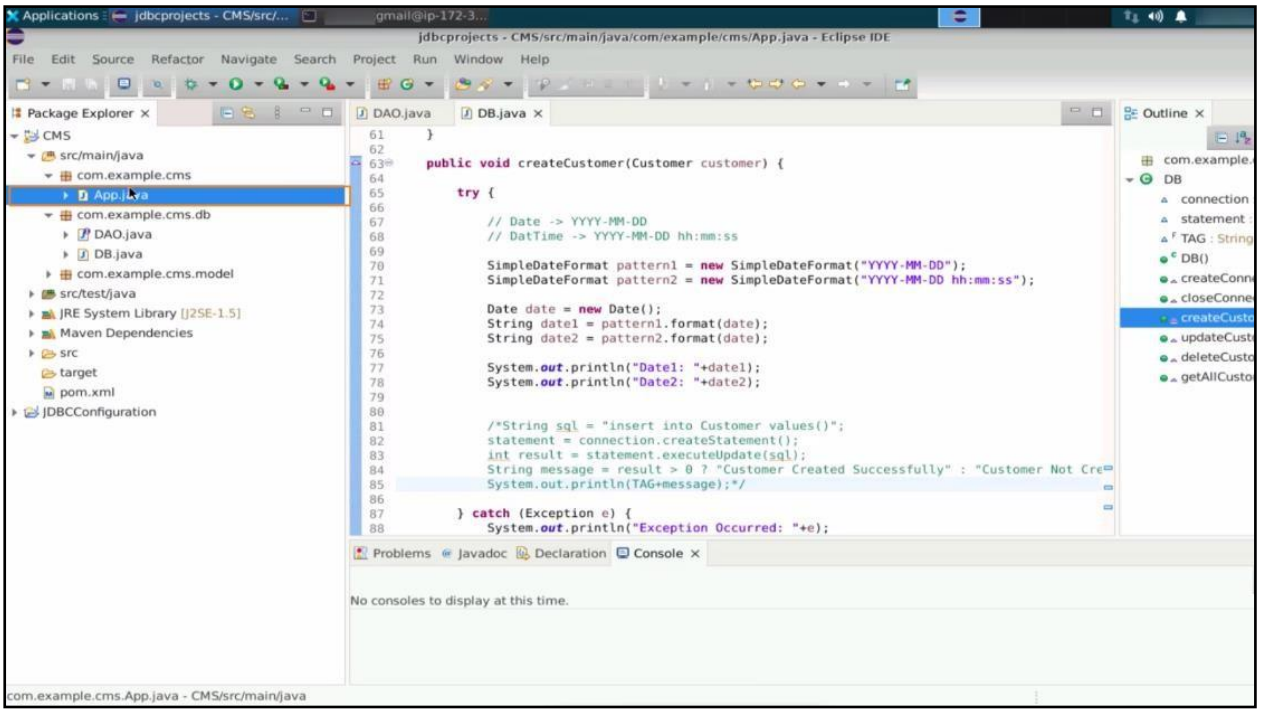
2.3 Create a simple **date** format API from line 65



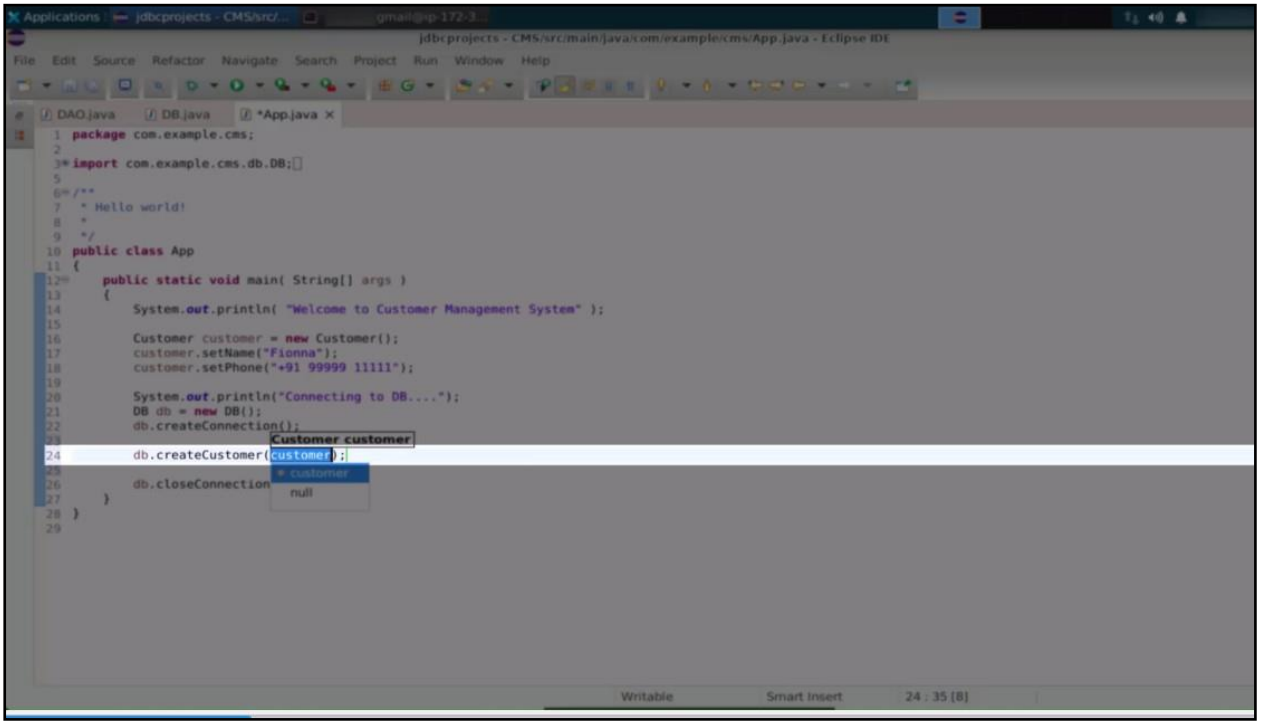
2.4 Comment out the following code:



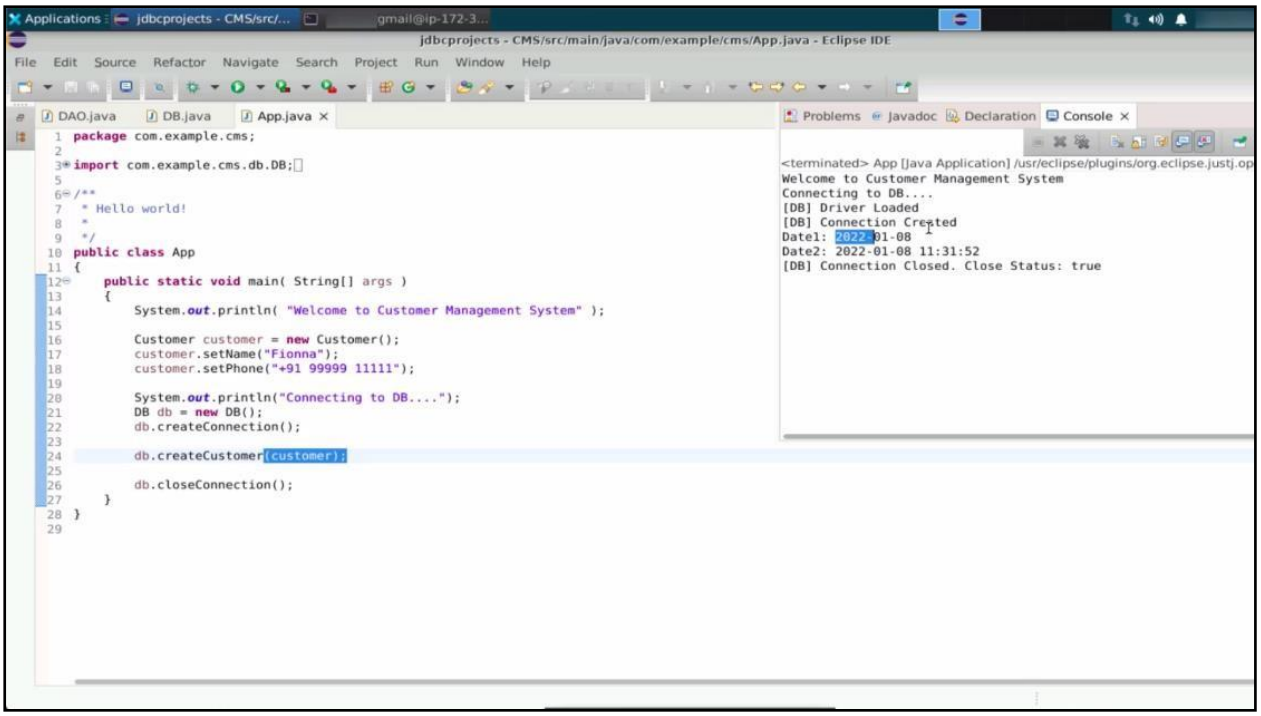
2.5 Navigate to the **App.java** file



2.6 Call the **db.createCustomer()** method

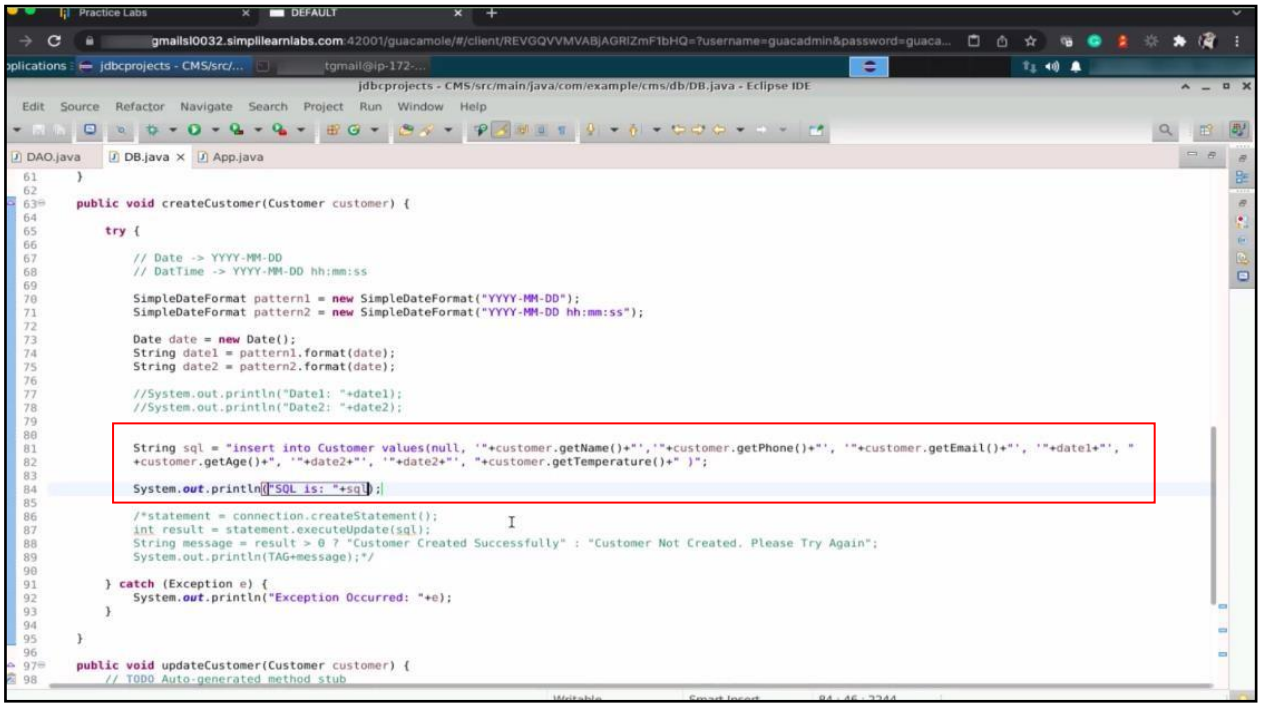


2.7 Run the code to get the output as **Connection Created** and the dates

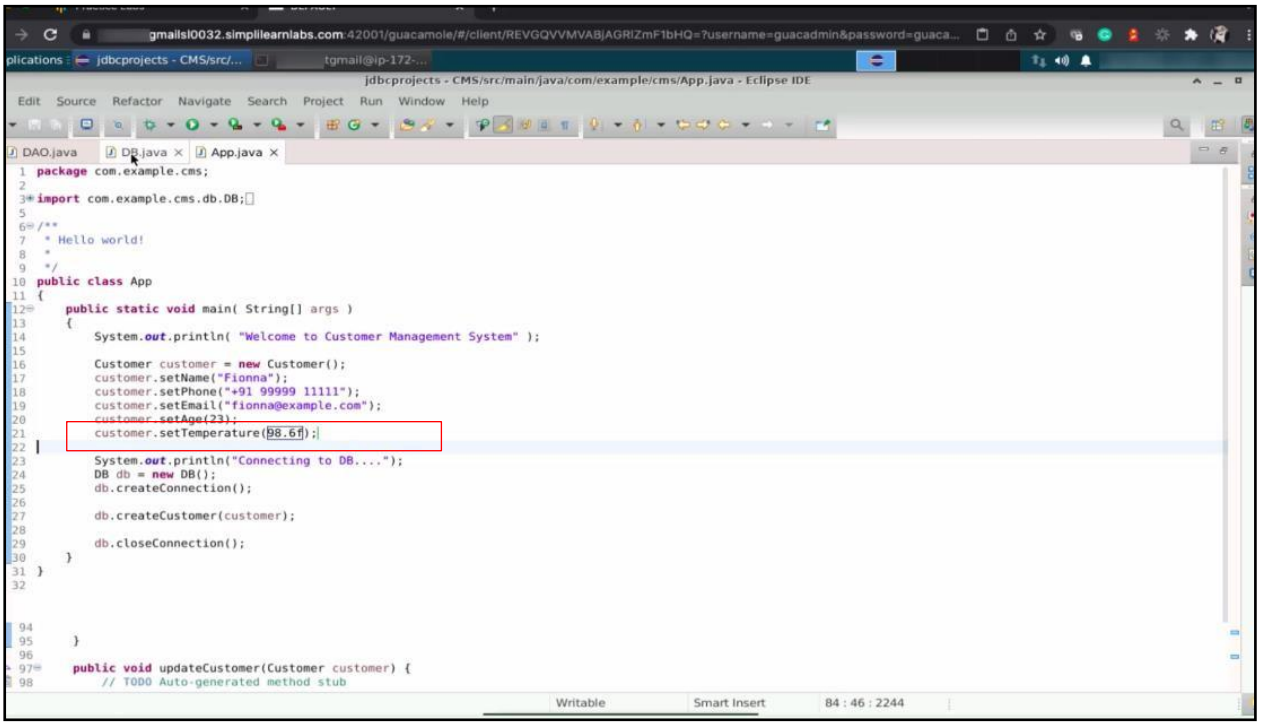


Step 3: Create a SQL query for the insert operation

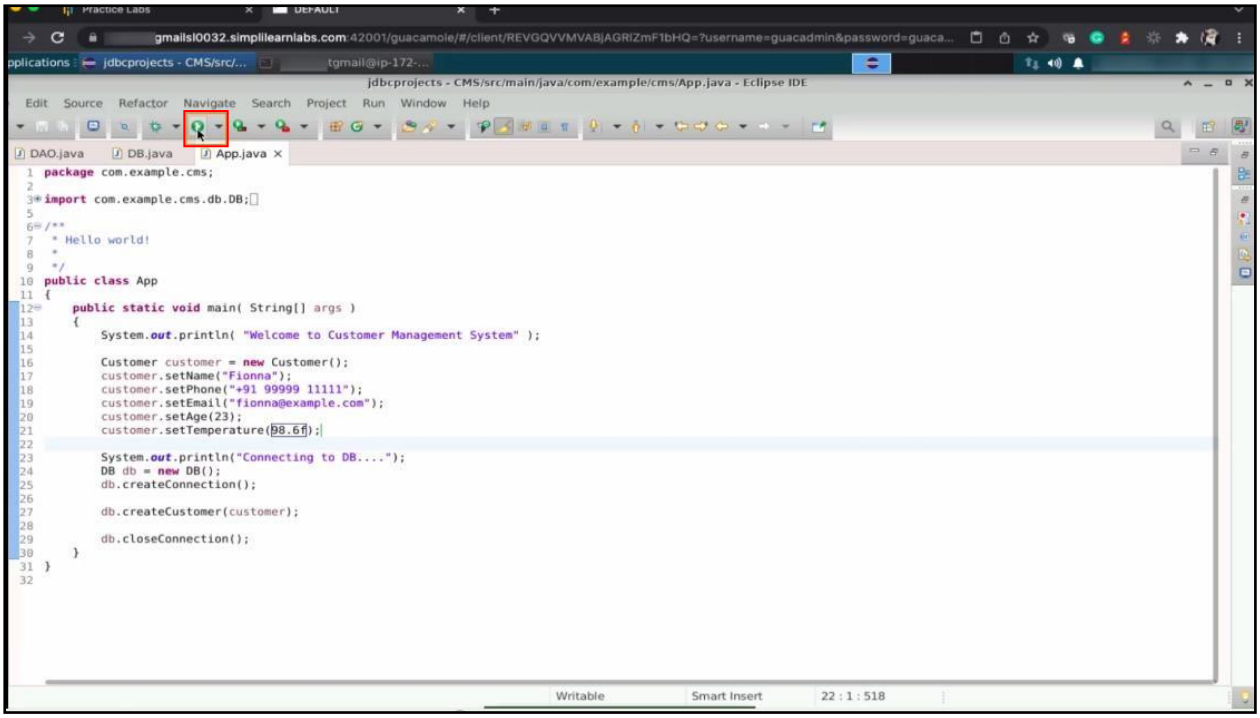
3.1 Create a SQL query to perform the insert operation and enter details into the Customer table



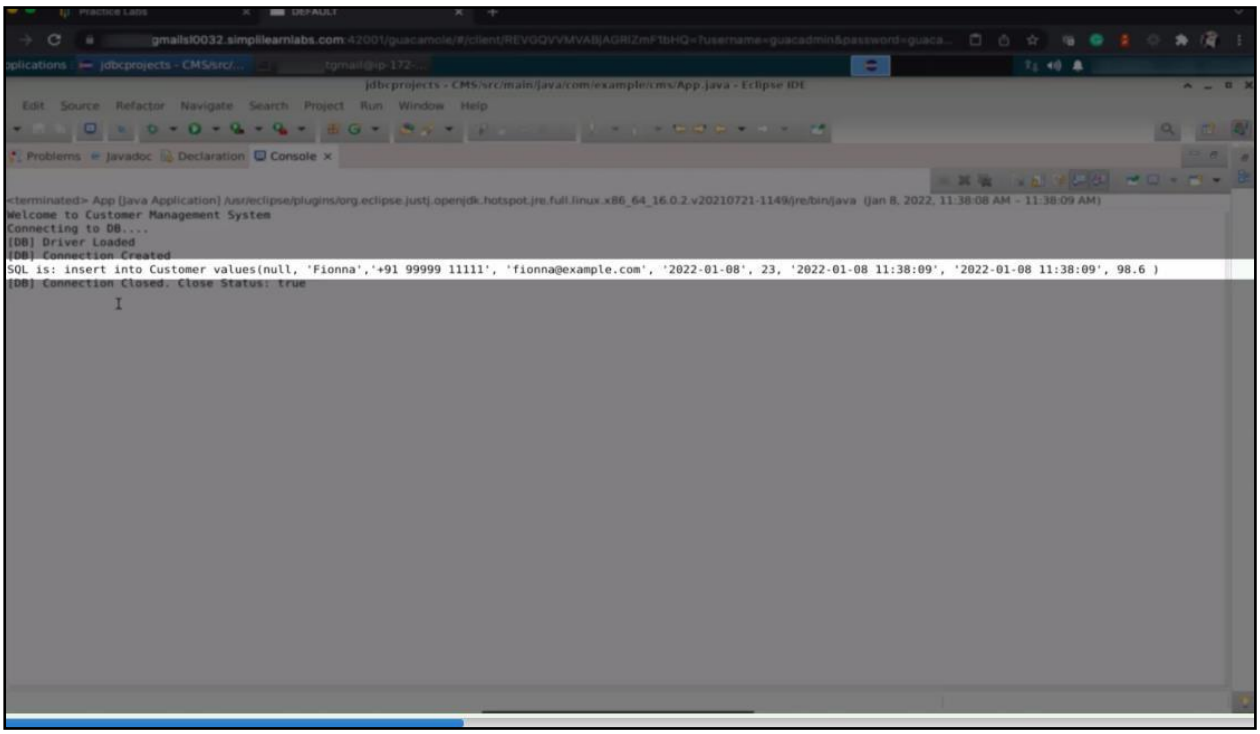
3.2 Return to the **App.java** file and add more attributes like **age** and **temperature**



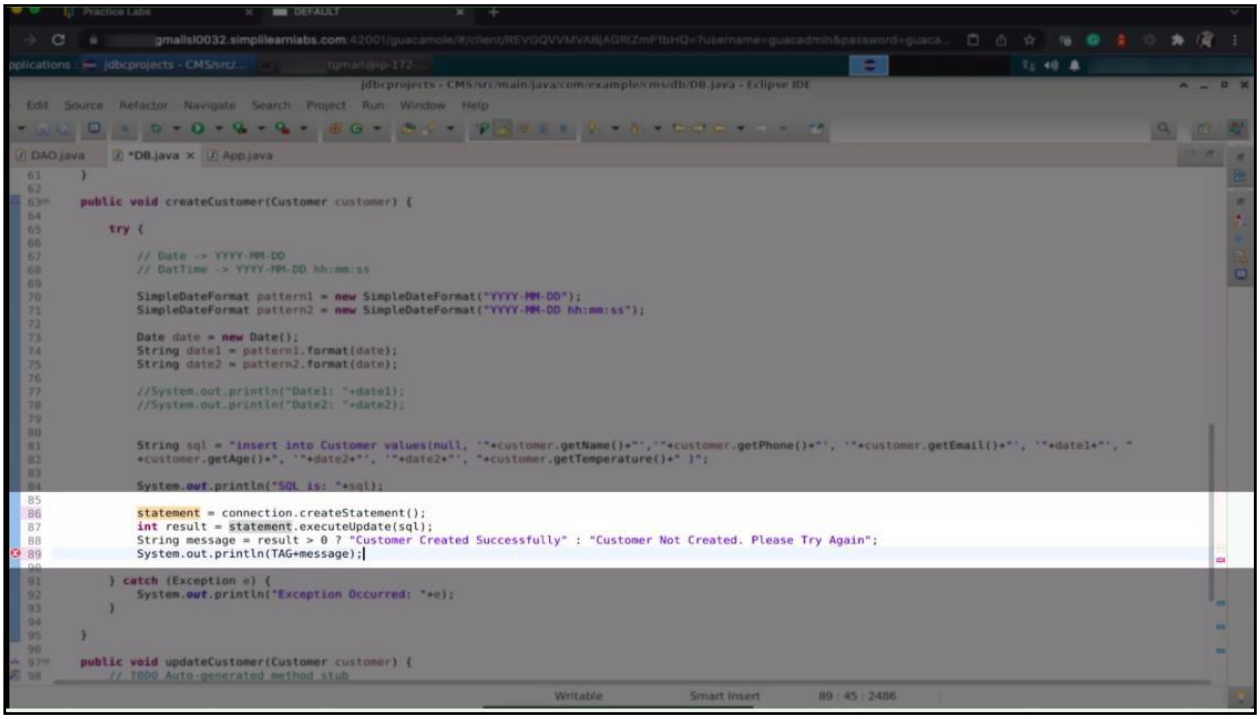
3.3 Save and run the code



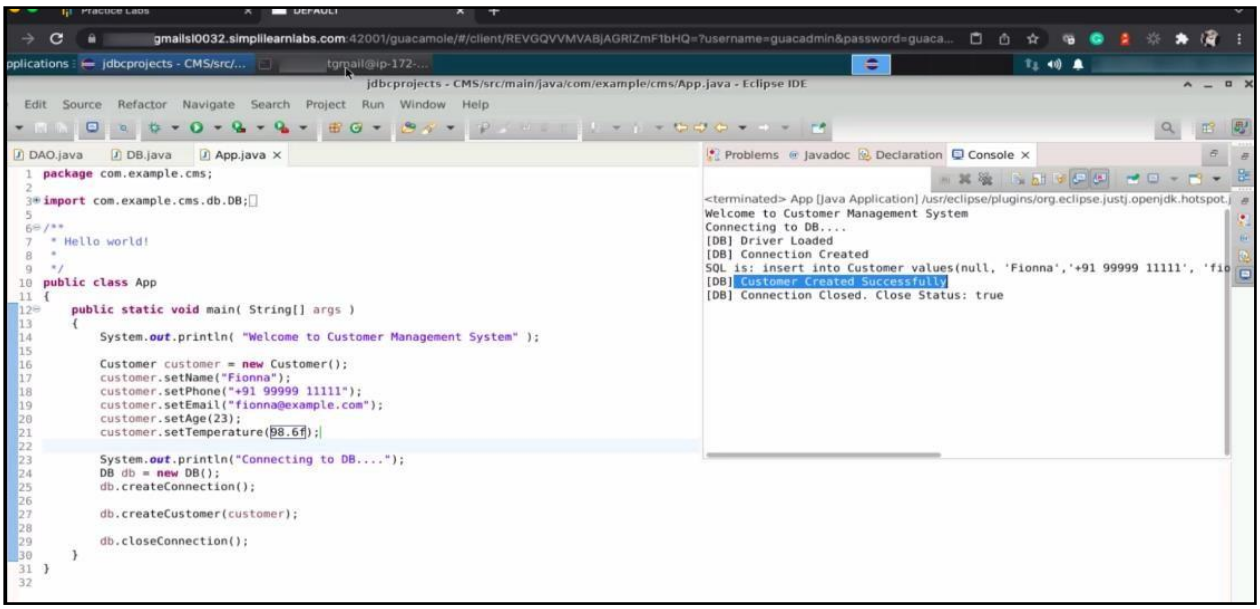
The SQL statement will be printed as written in the print statement.



3.4 Navigate to the **DB.java** file and uncomment the connection string part to establish a connection to the database for further operations:

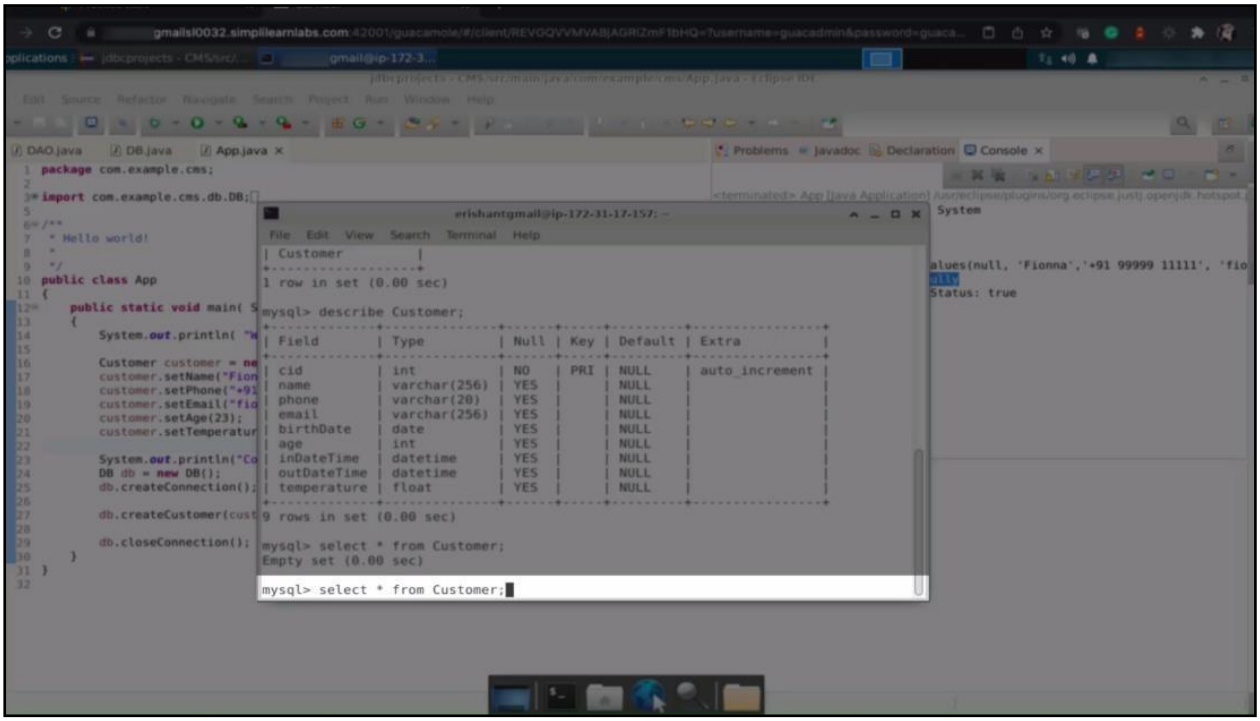


3.5 Run the code and observe the output **Customer Created Successfully** confirming the successful creation of the customer.

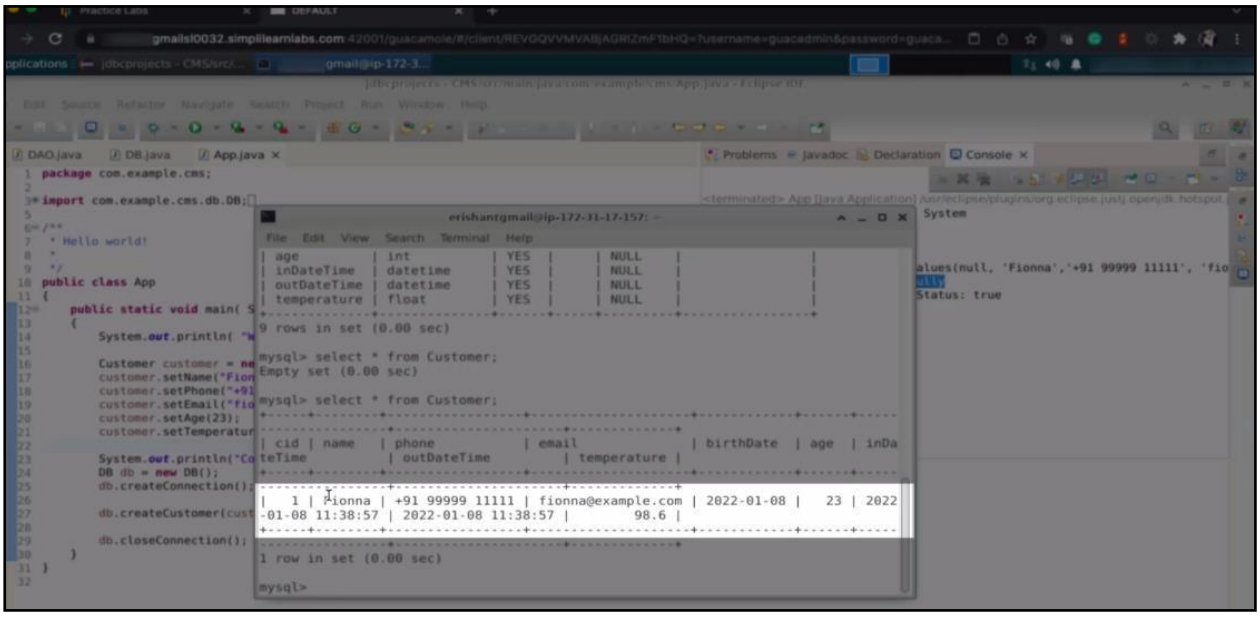


3.6 Go to the terminal and run the select command:

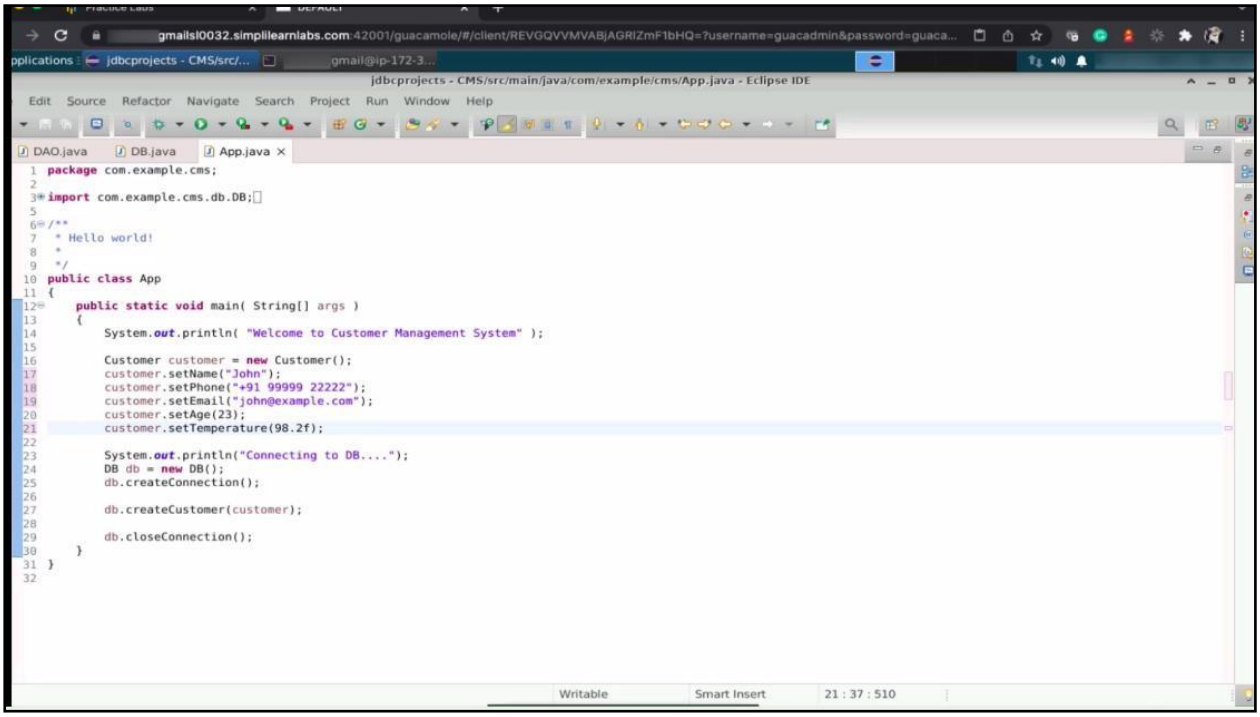
select * from Customer;



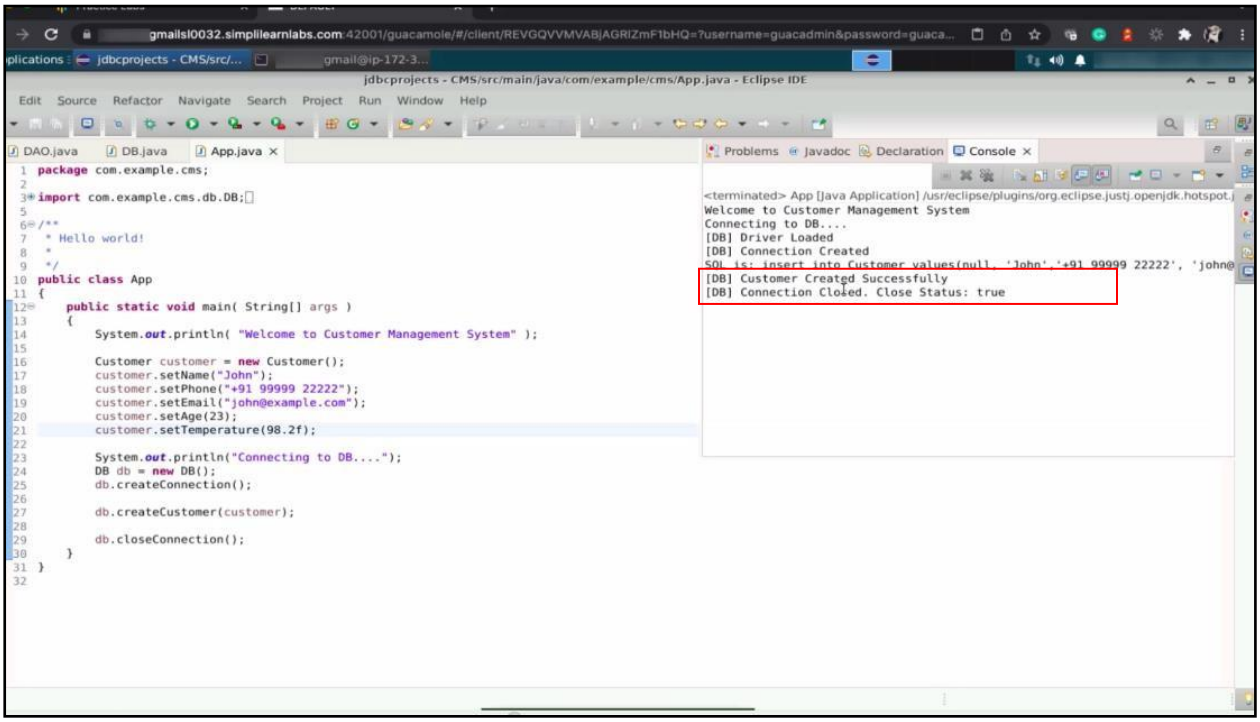
You can view the inserted data in the table **Customer**



3.7 Insert another record in the Customer table using the **set()** operation

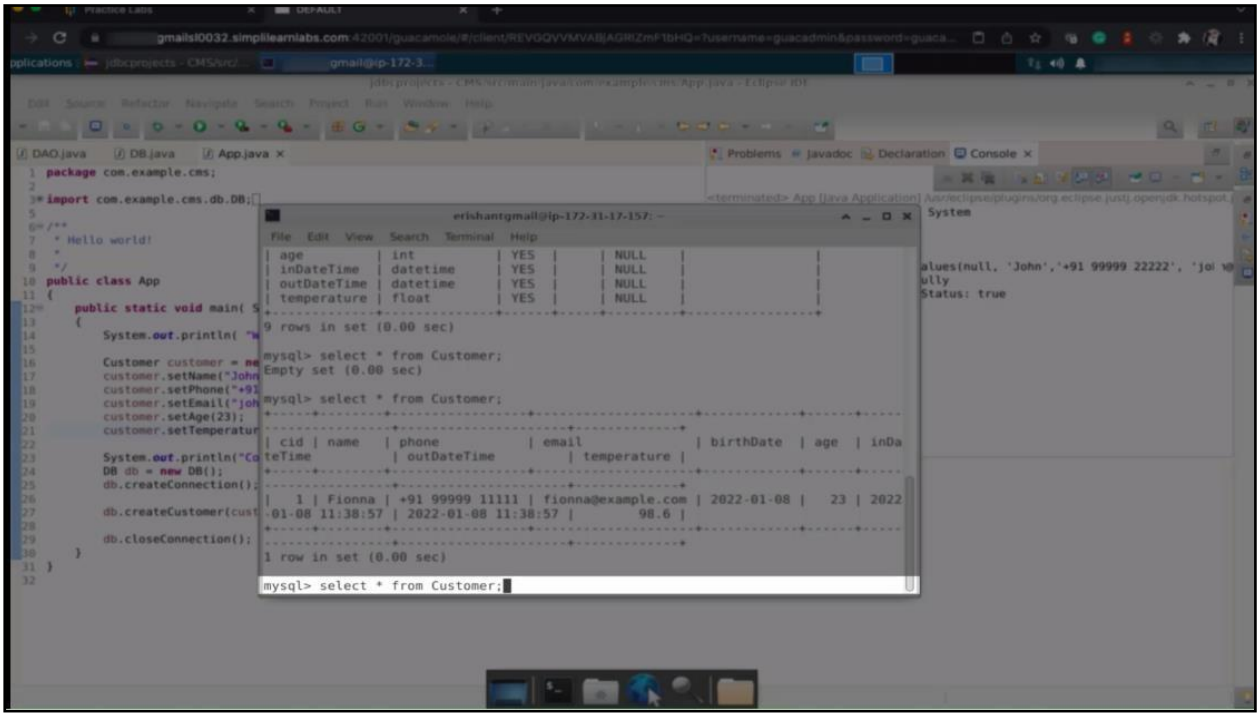


3.8 Run the code to get the output as **Customer Created Successfully**

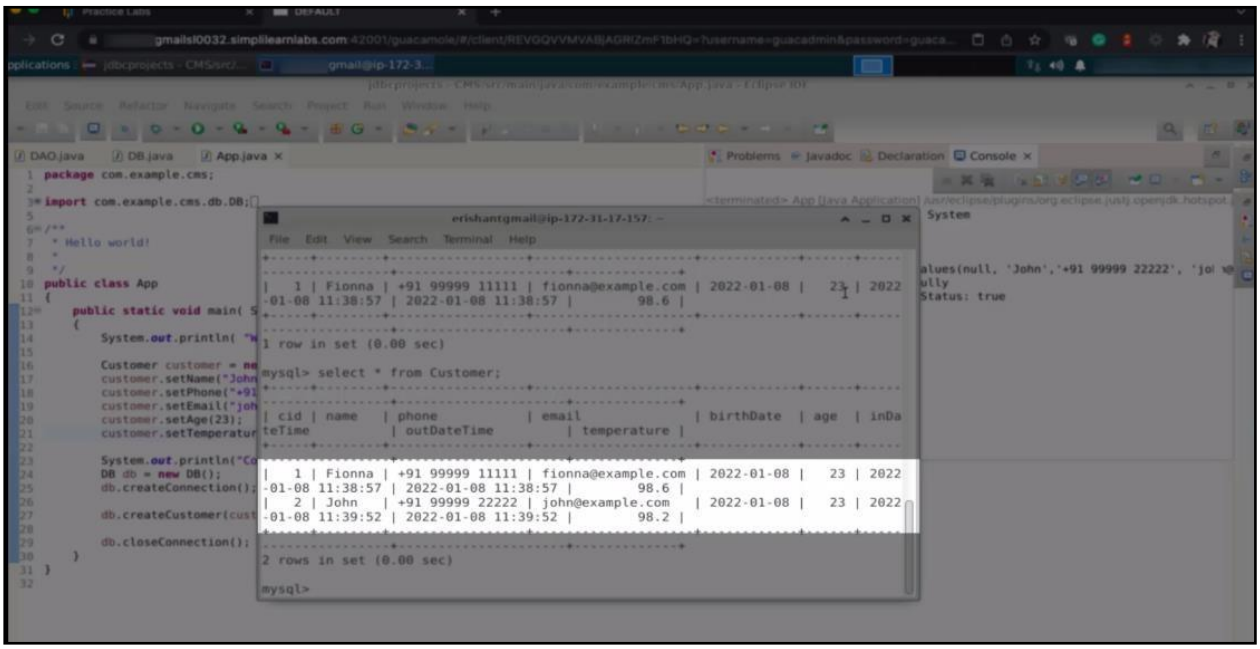


3.9 Return to the terminal and rerun the select command:

select * from Customer;

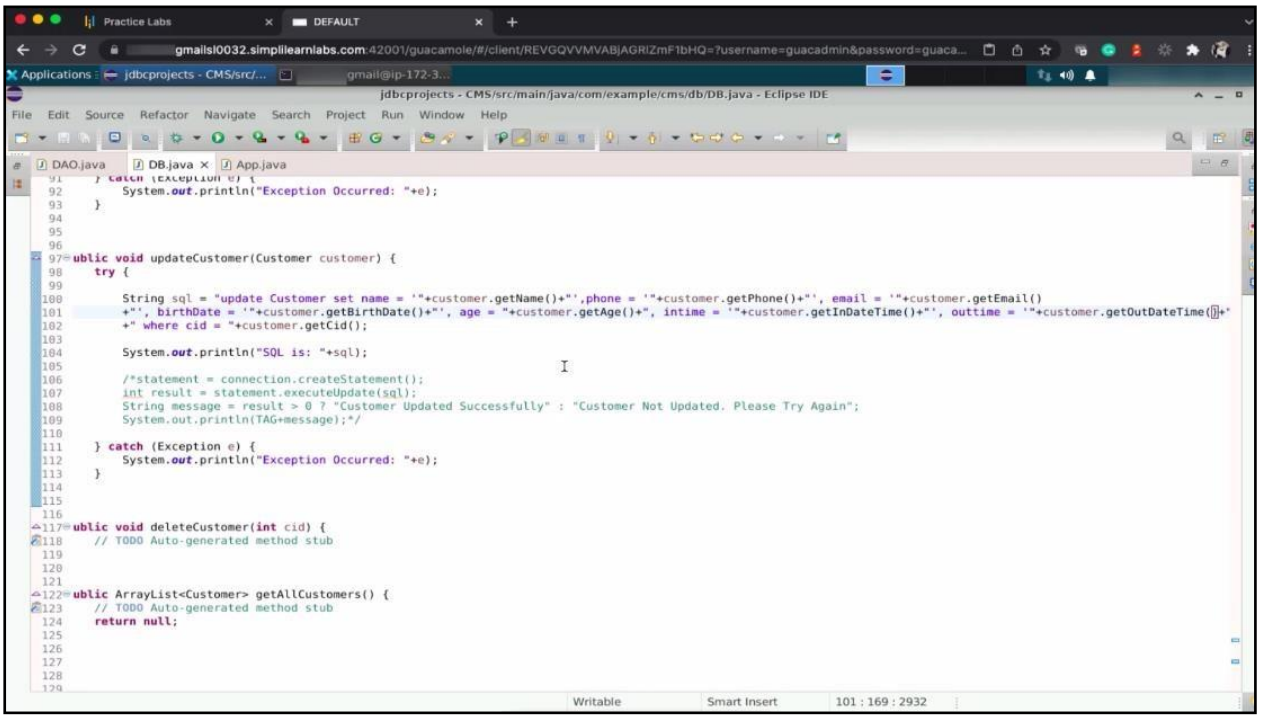


You will see another record is inserted in the table.

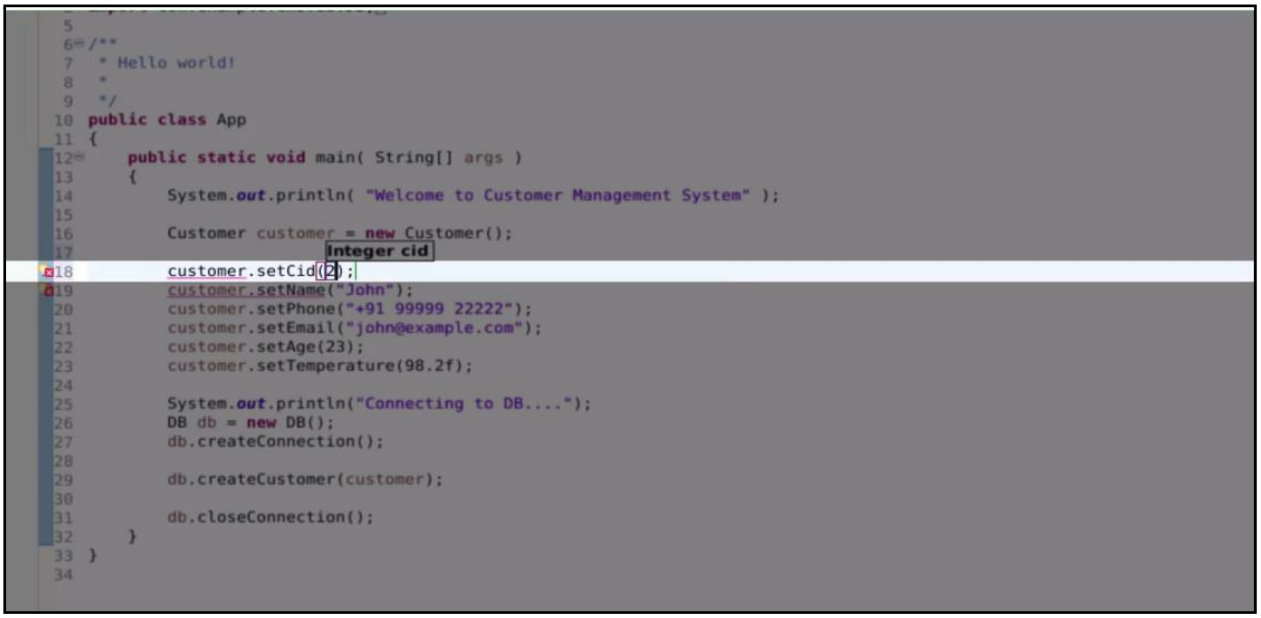


Step 4: Write an update operation

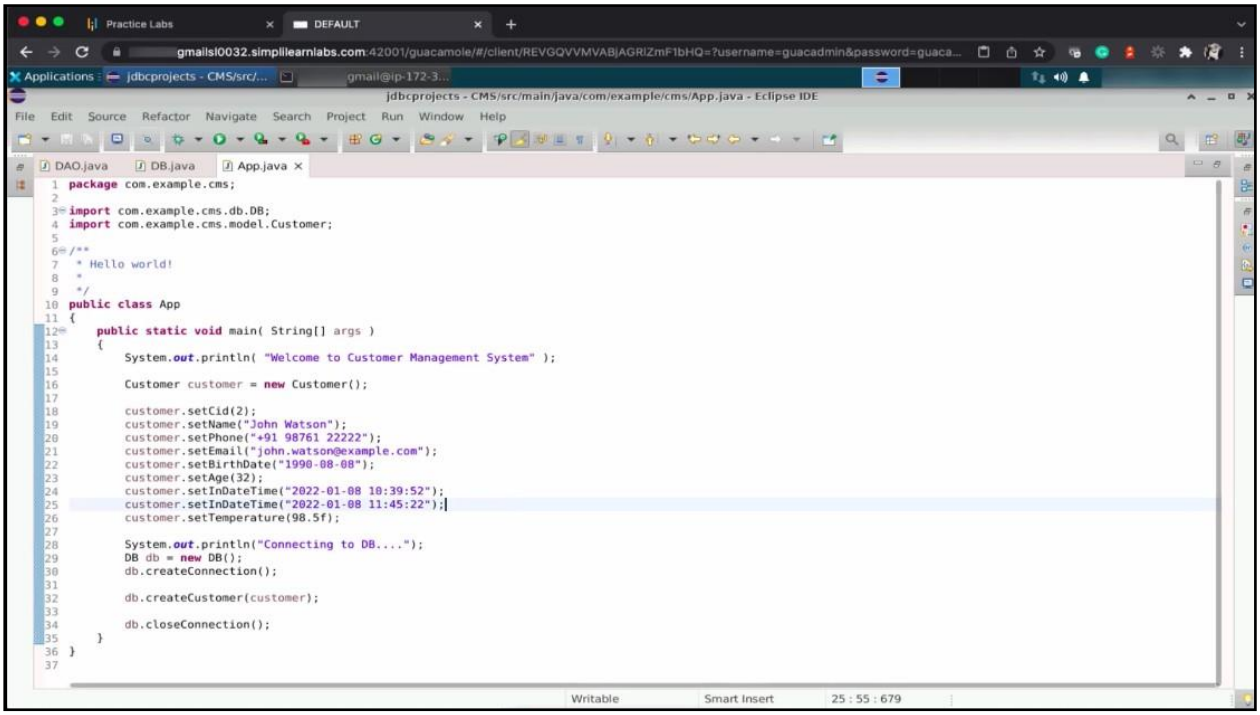
4.1 Write an update operation in the **try-catch** block



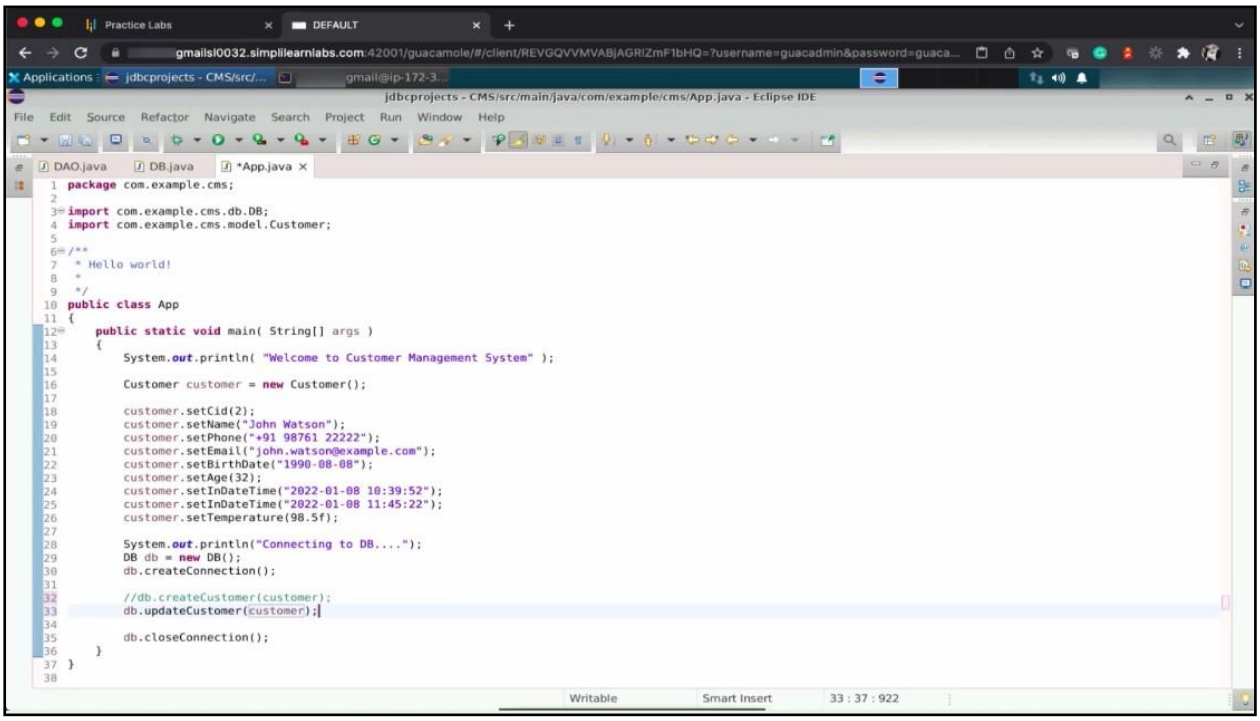
4.2 Return to the **App.java** file and set the **customer id**



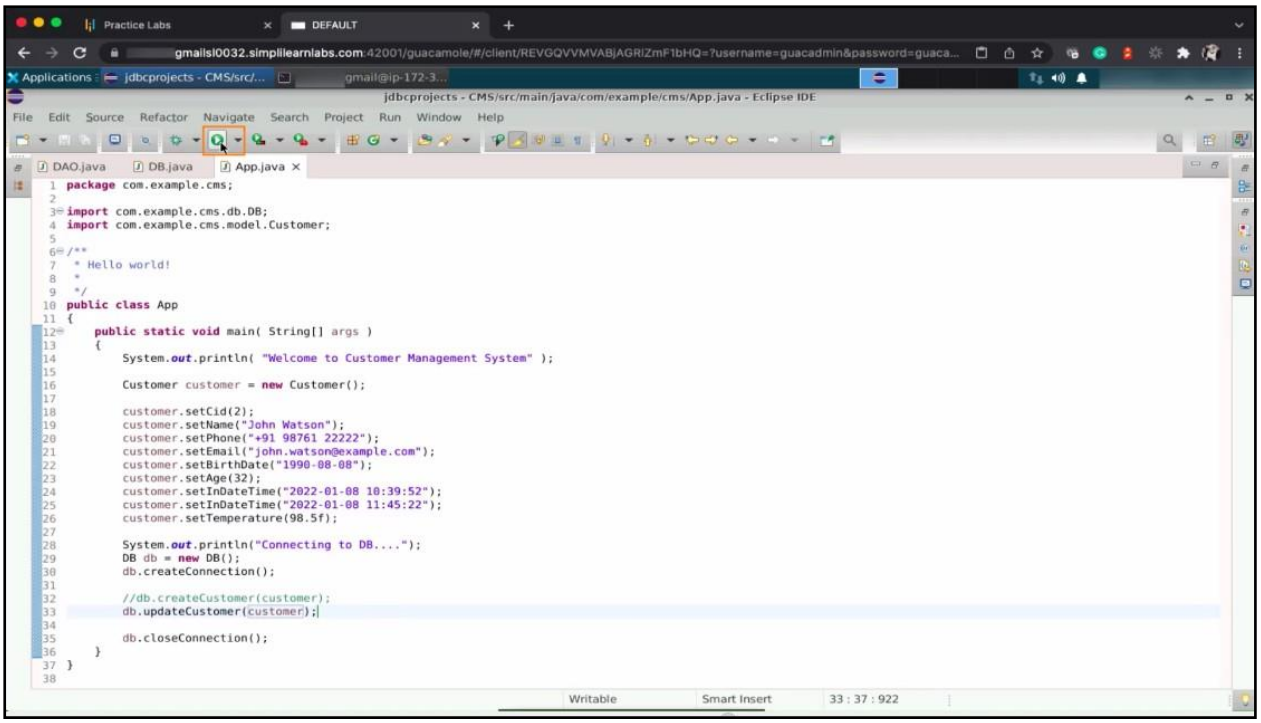
4.3 Update all the details of customers using the **set()** method



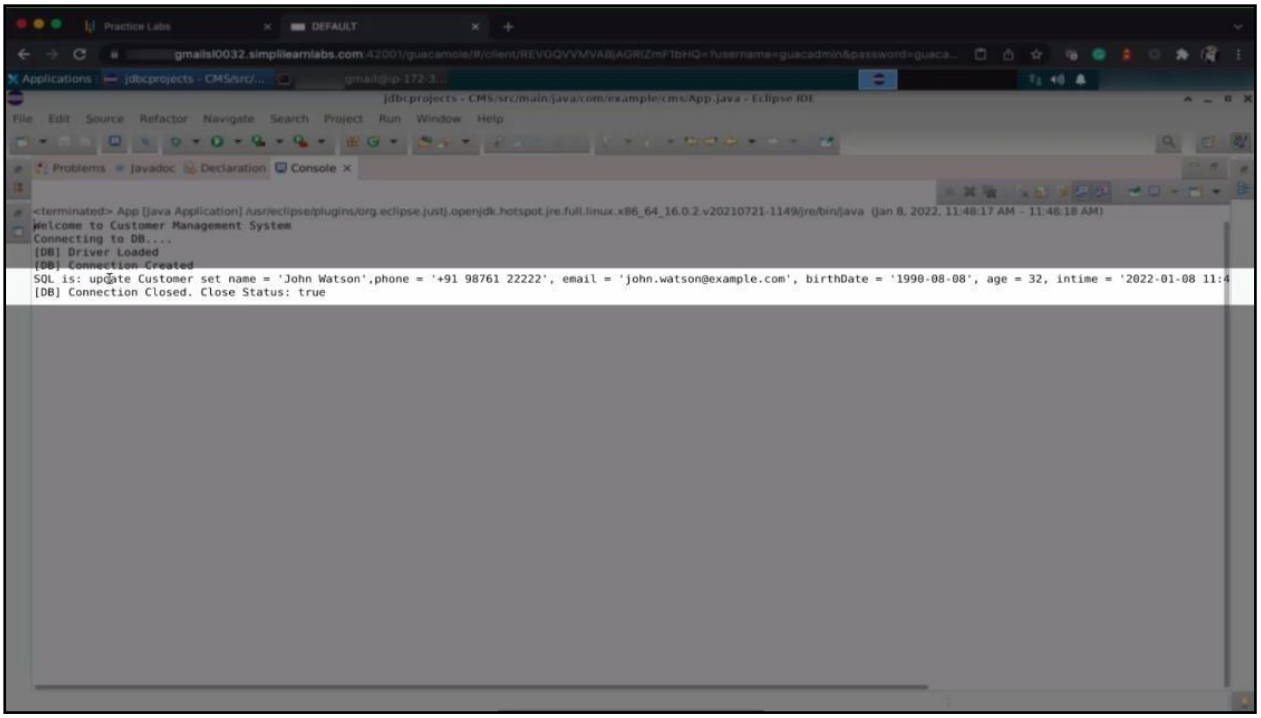
4.4 Write the **db.updateCustomer()** method for updating details



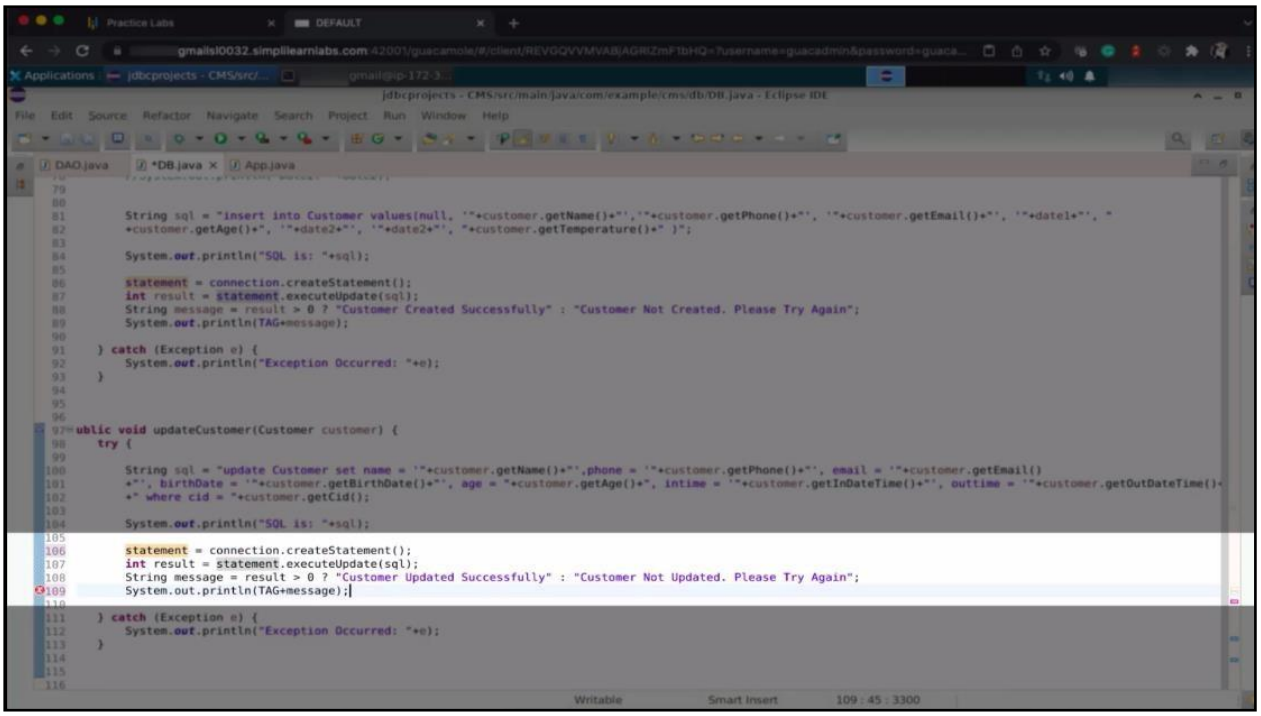
4.5 Save and run the code



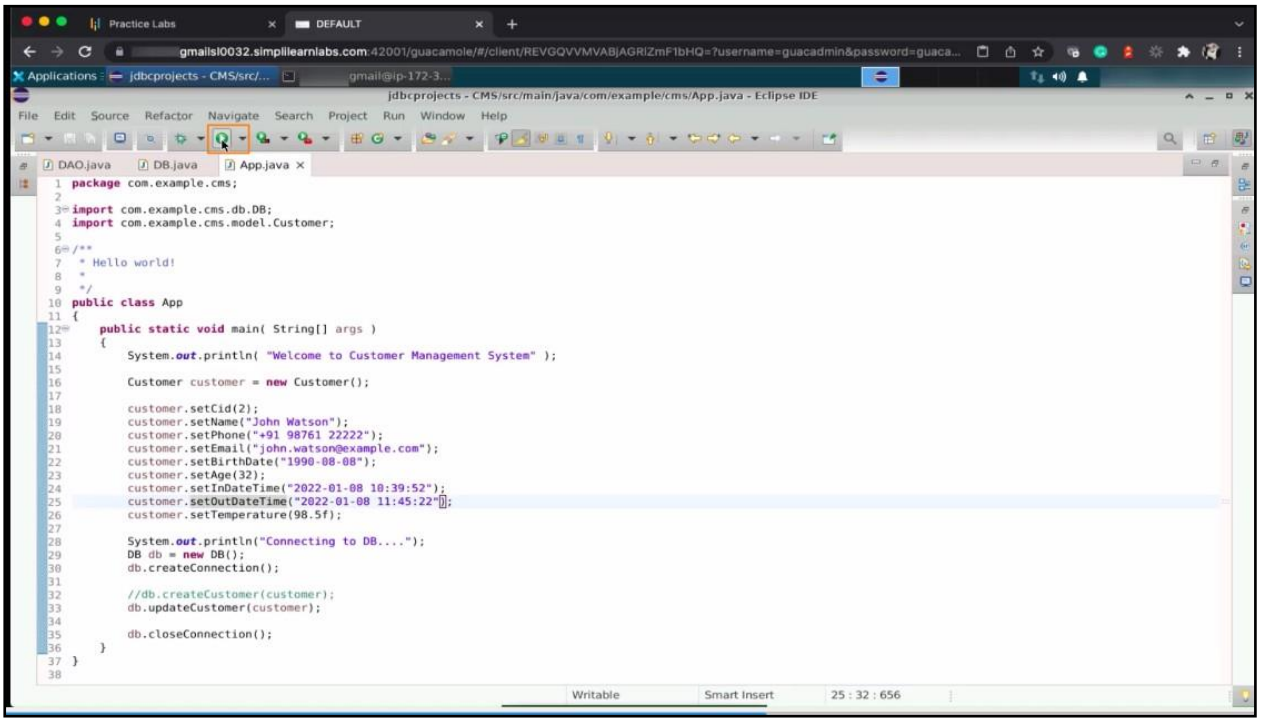
You will see that the customer details are updated.



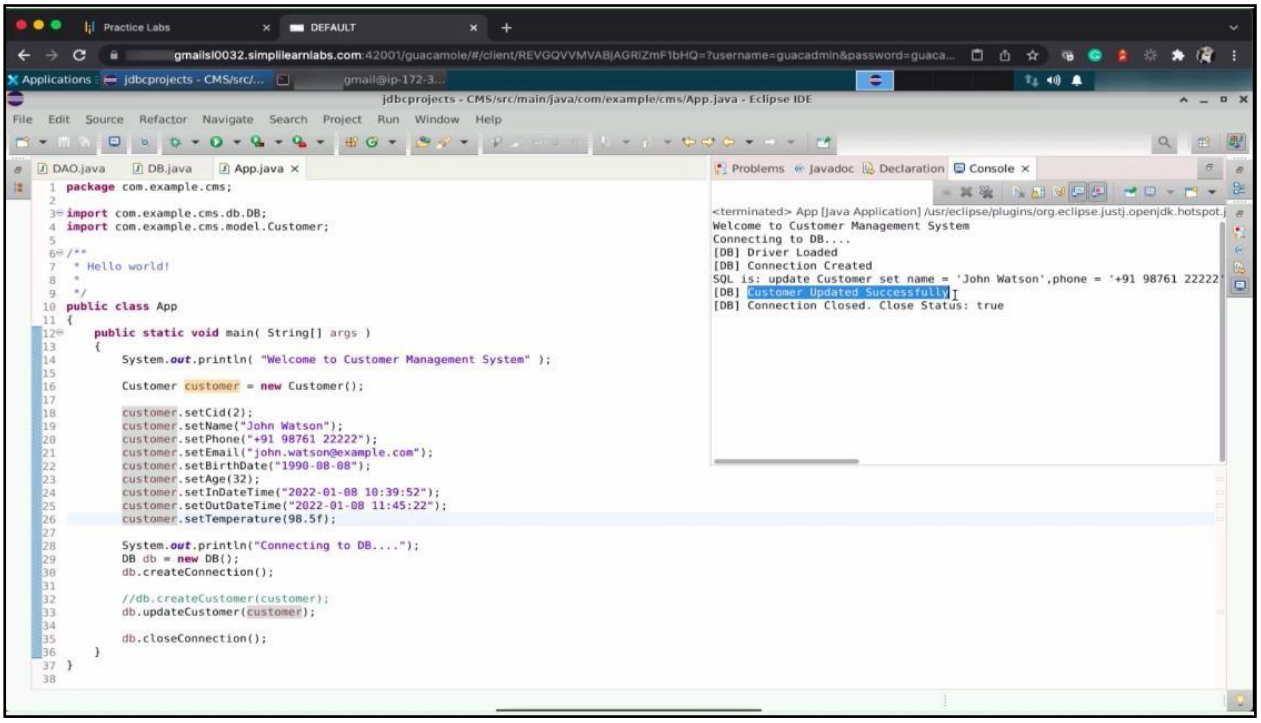
4.6 Return to the **DB.java** file and uncomment the connection string statement



4.7 Save and run the code

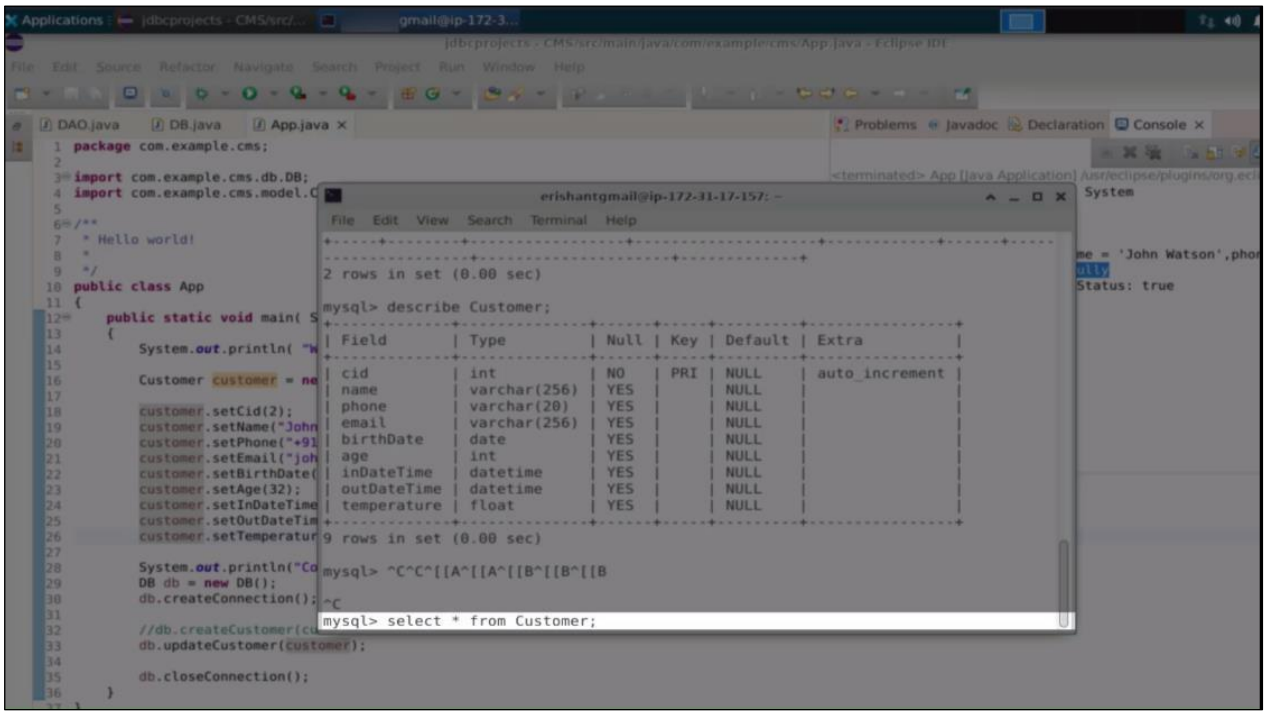


After the update operation is completed successfully, the output **Customer Updated Successfully** will be shown.

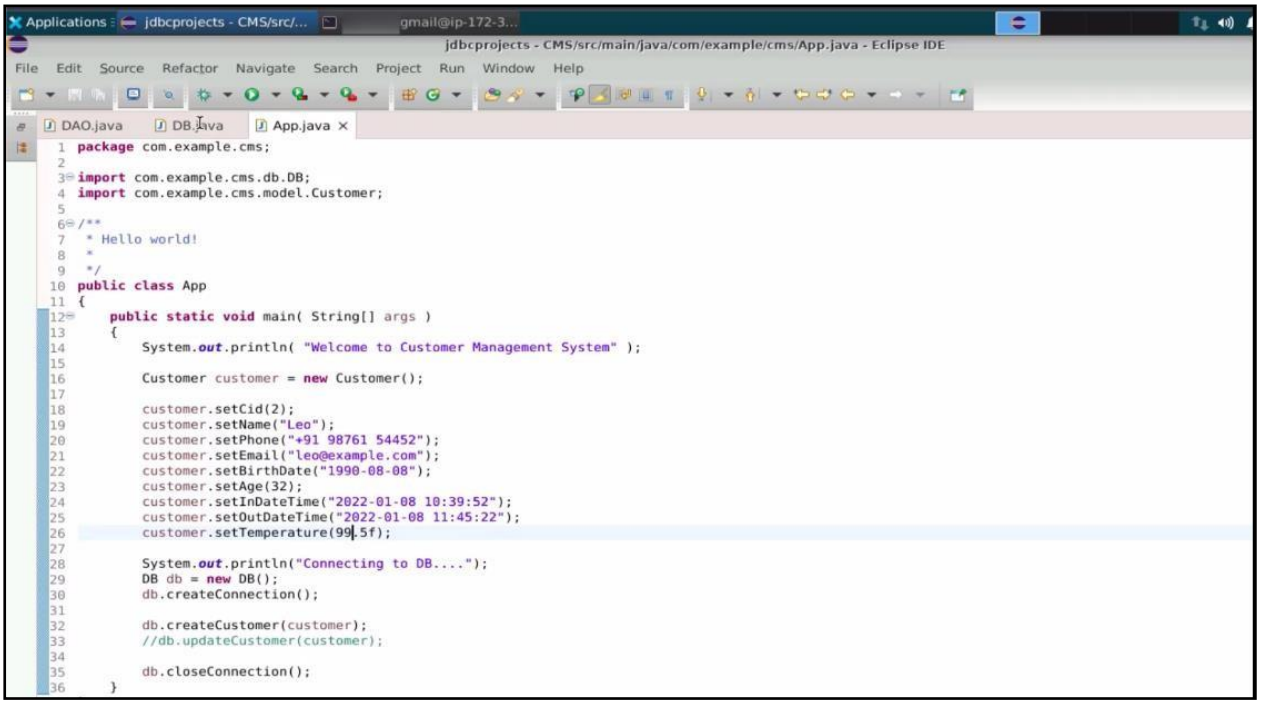


4.8 Return to the terminal and run the **select** command:

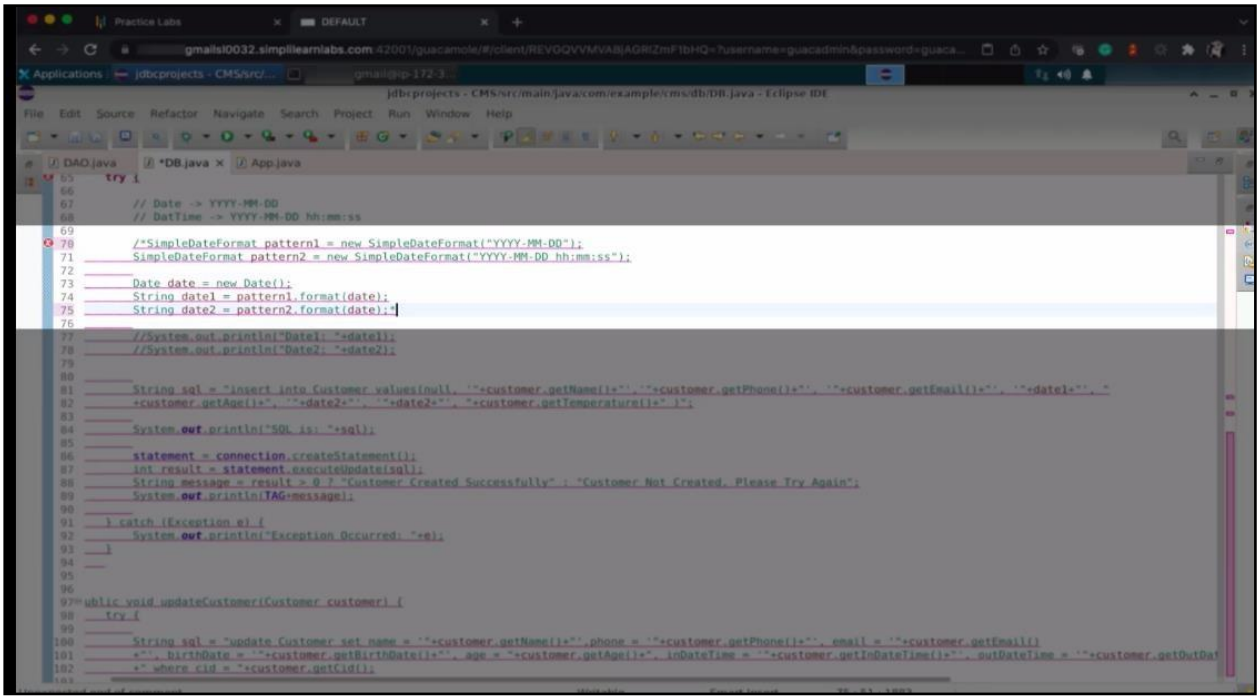
select * from Customer



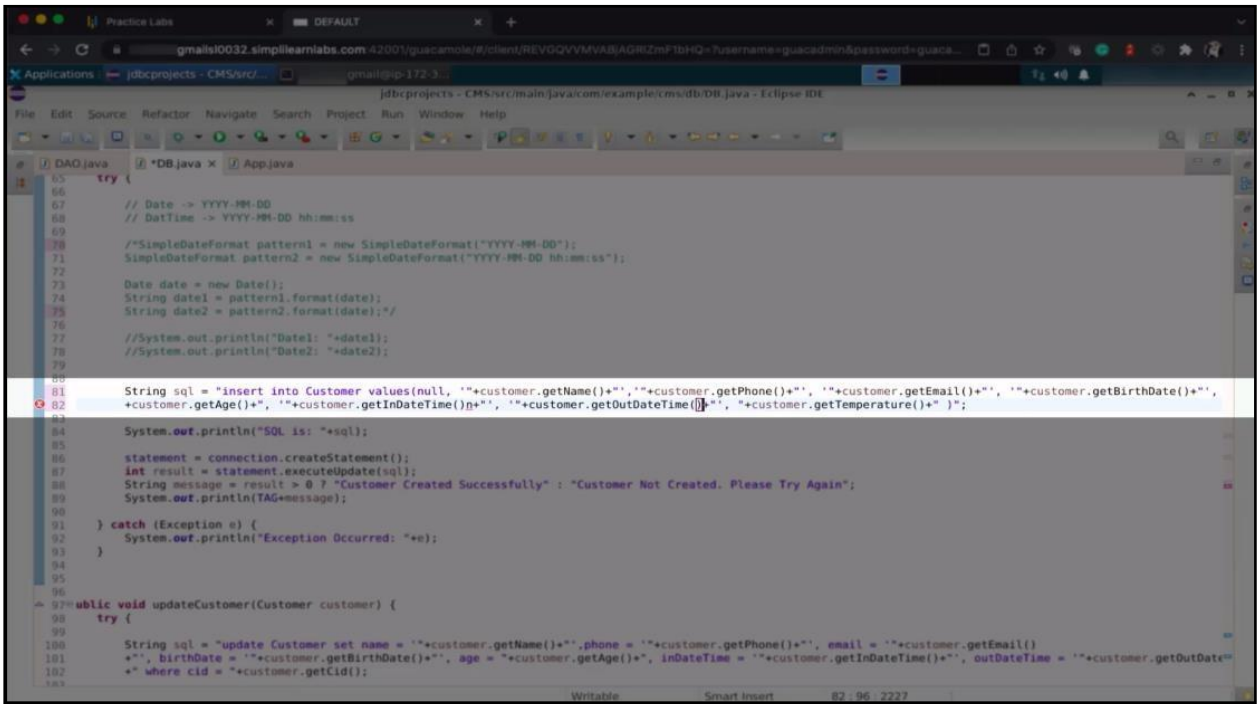
4.9 Insert one more customer detail by updating the details and running the code again in the **App.file** file



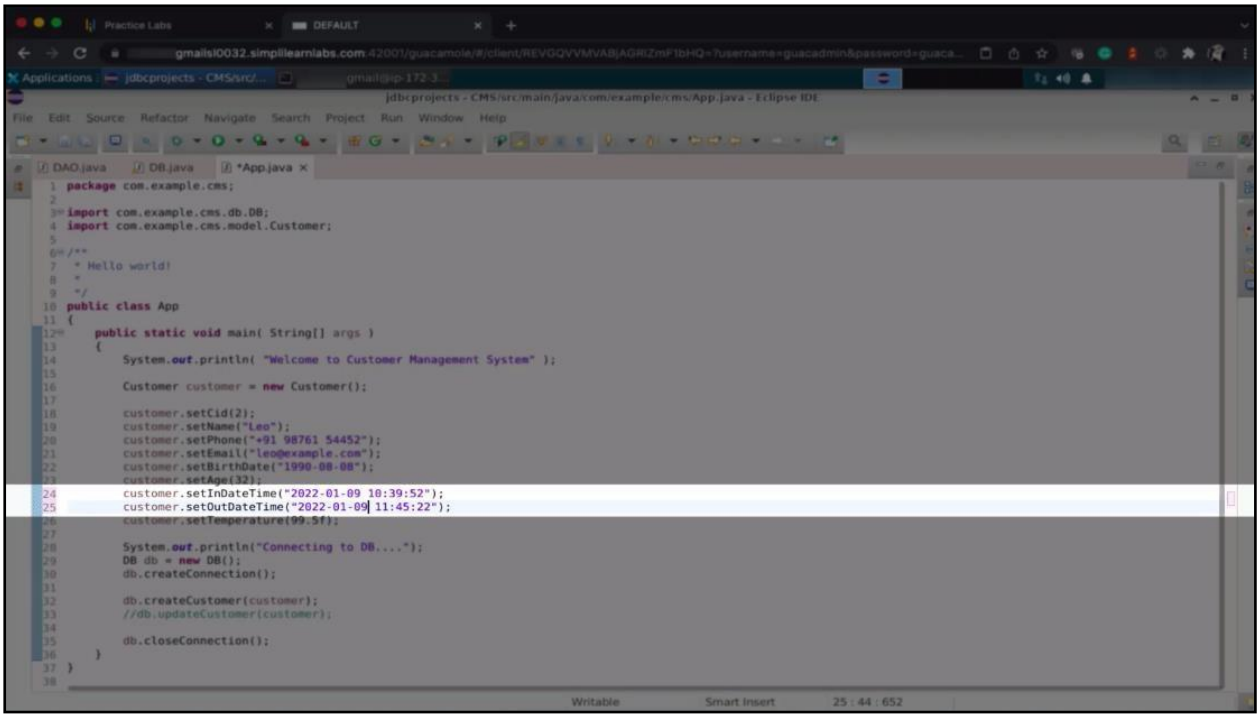
4.10 Return to the **DB.java** file and comment on the date format code to prevent potential errors in the future if the entered format doesn't match the expected format



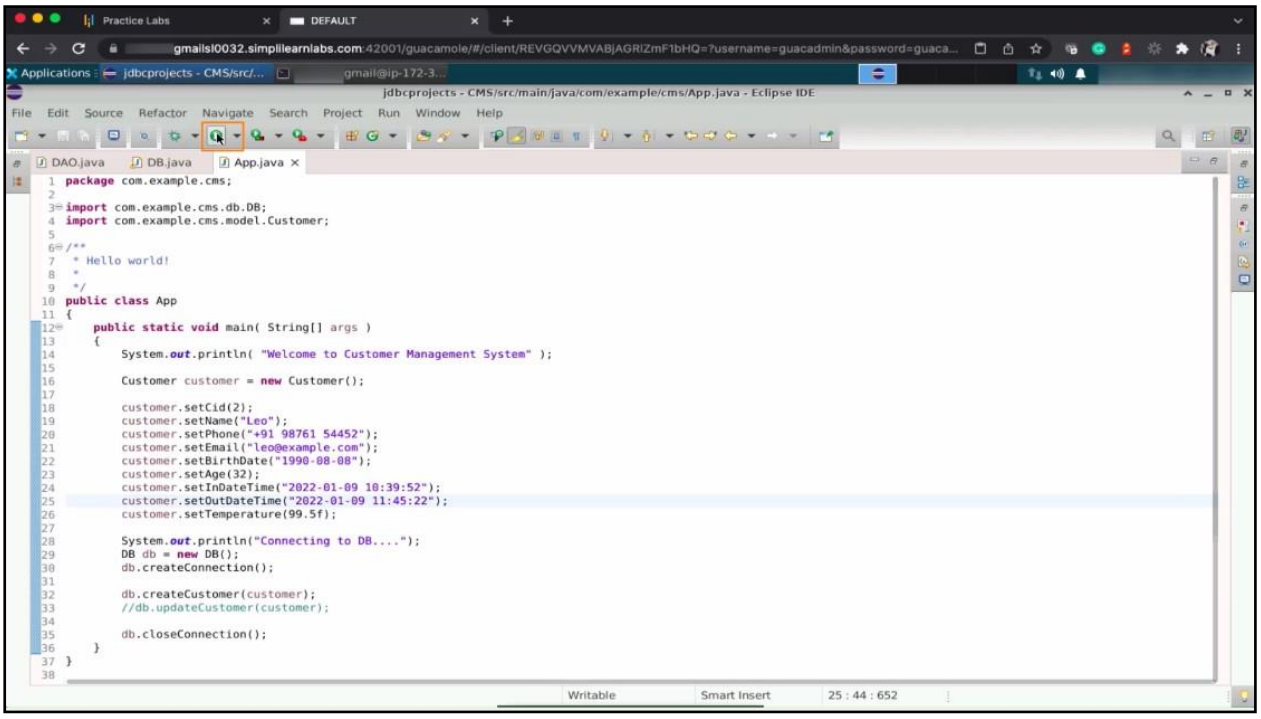
4.11 Change the query as shown:



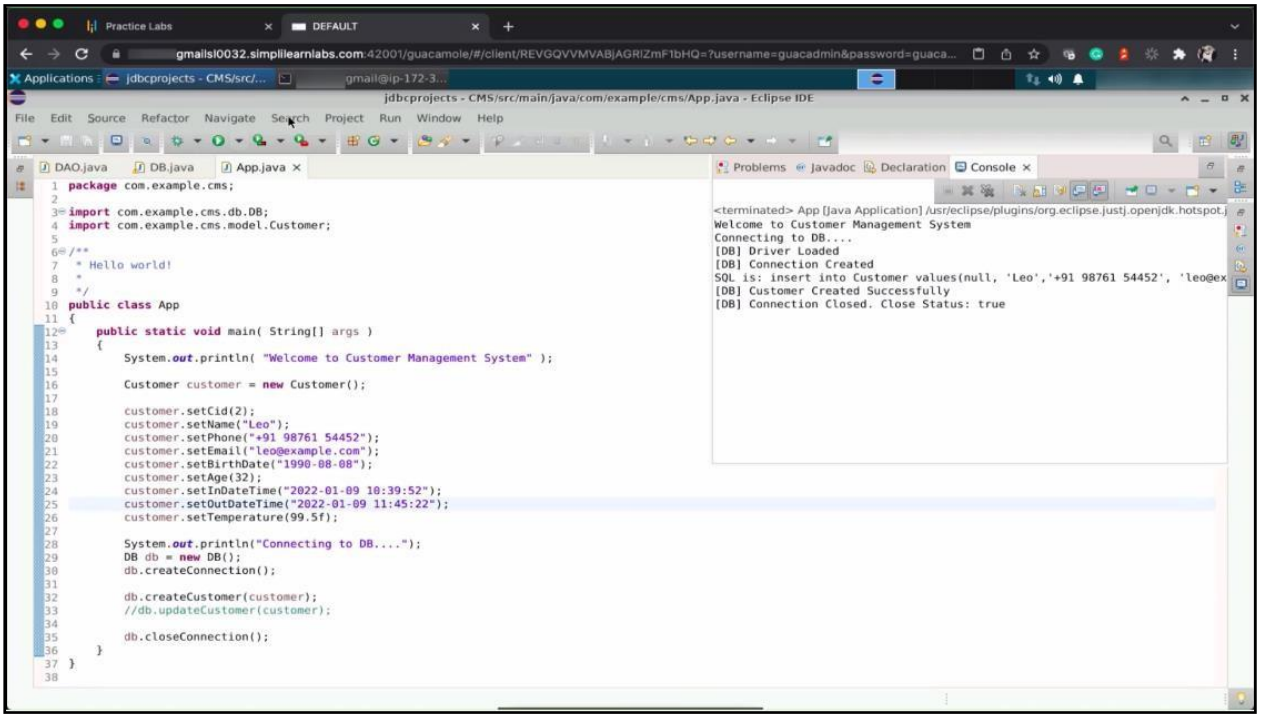
4.12 Return to the **App.java** and change the **setInDateTime** and the **setOutDateTime**



4.13 Save and run the code

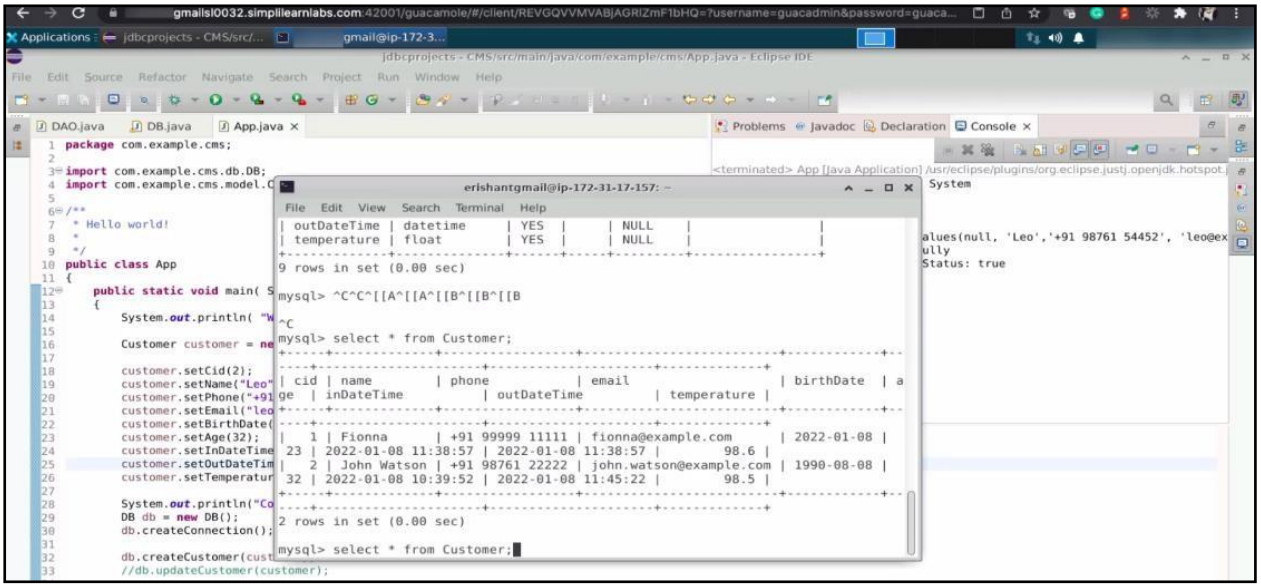


You can see the output as **Customer Created Successfully**.

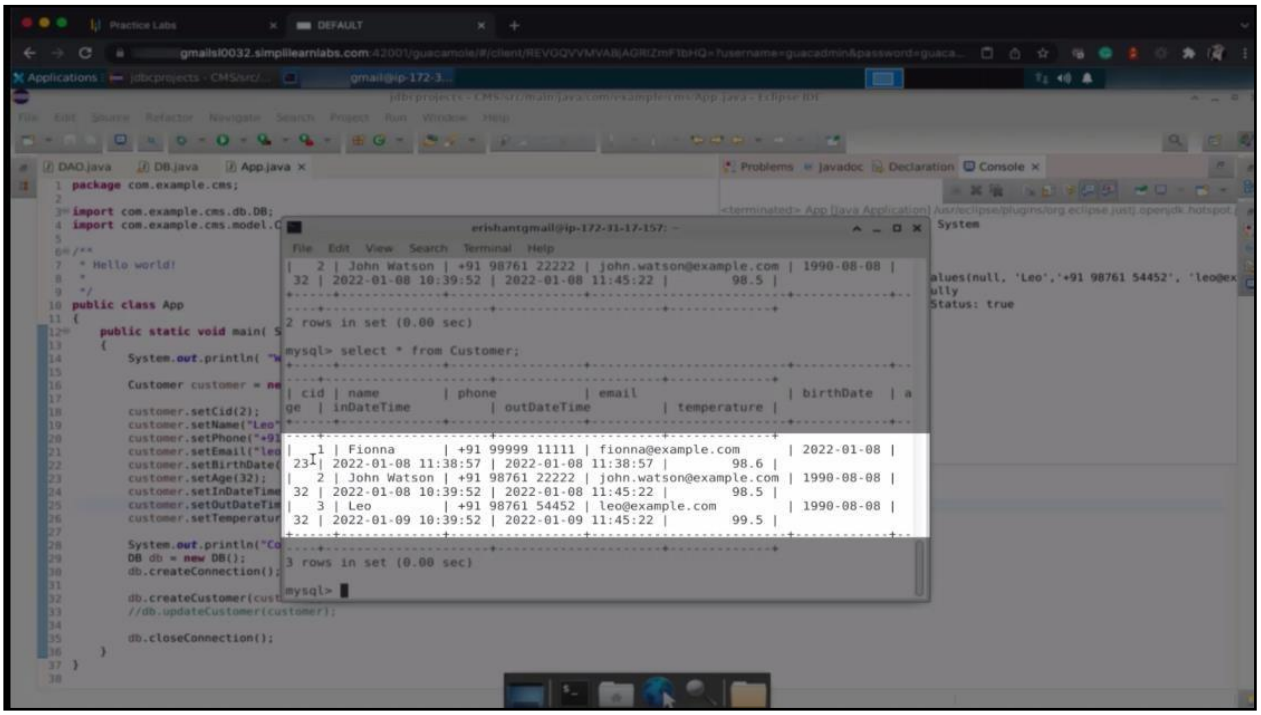


4.14 Go to the terminal and run the **select** command:

select * from Customers;

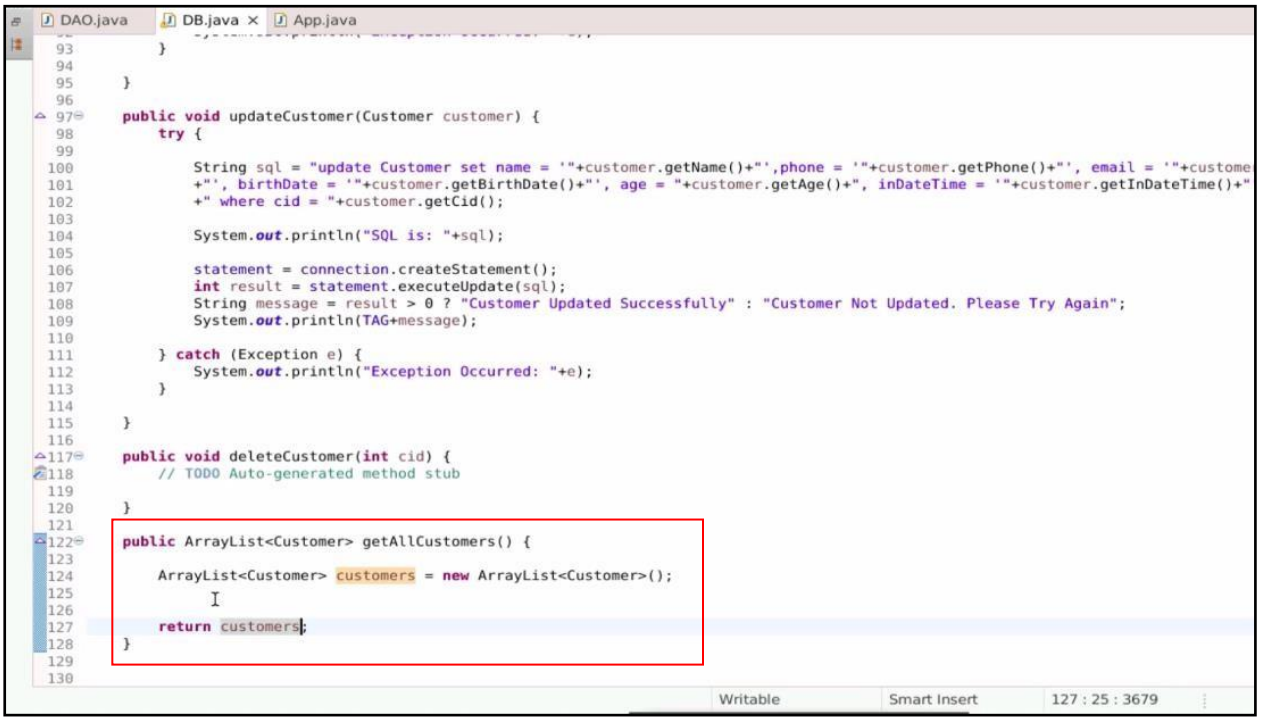


You can see the updated data in the table:

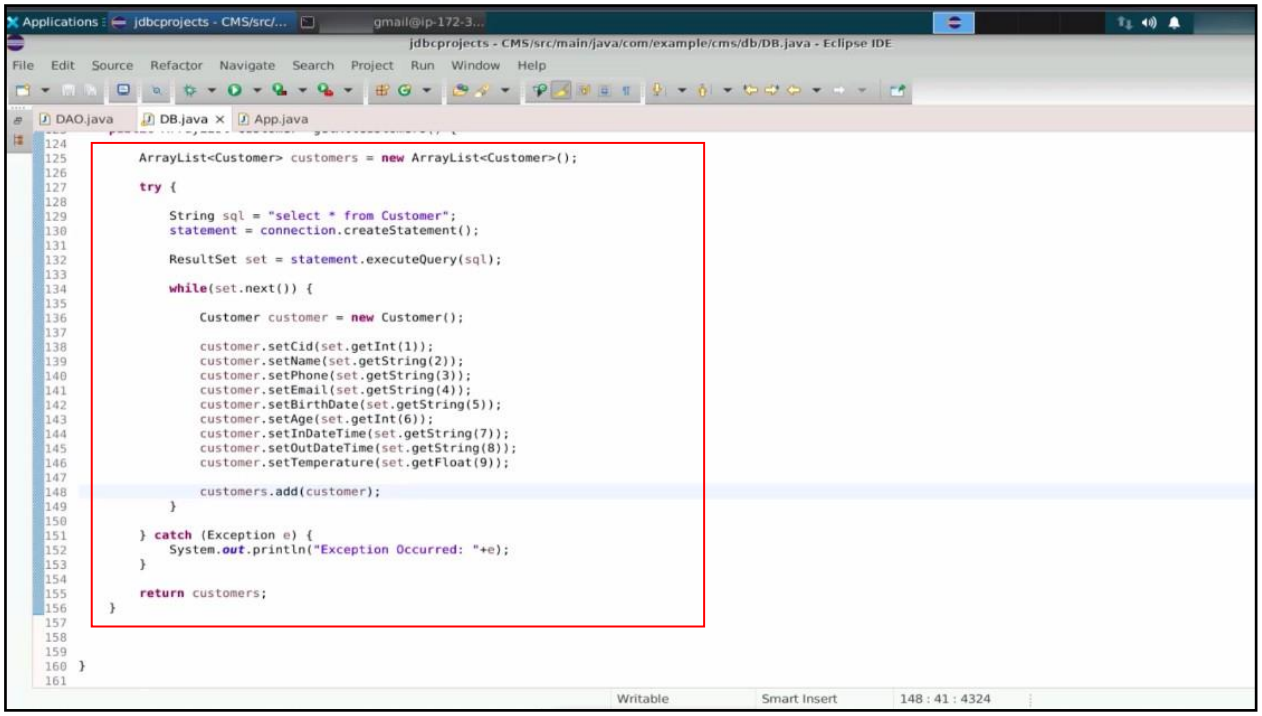


Step 5: Perform the getallcustomer operations

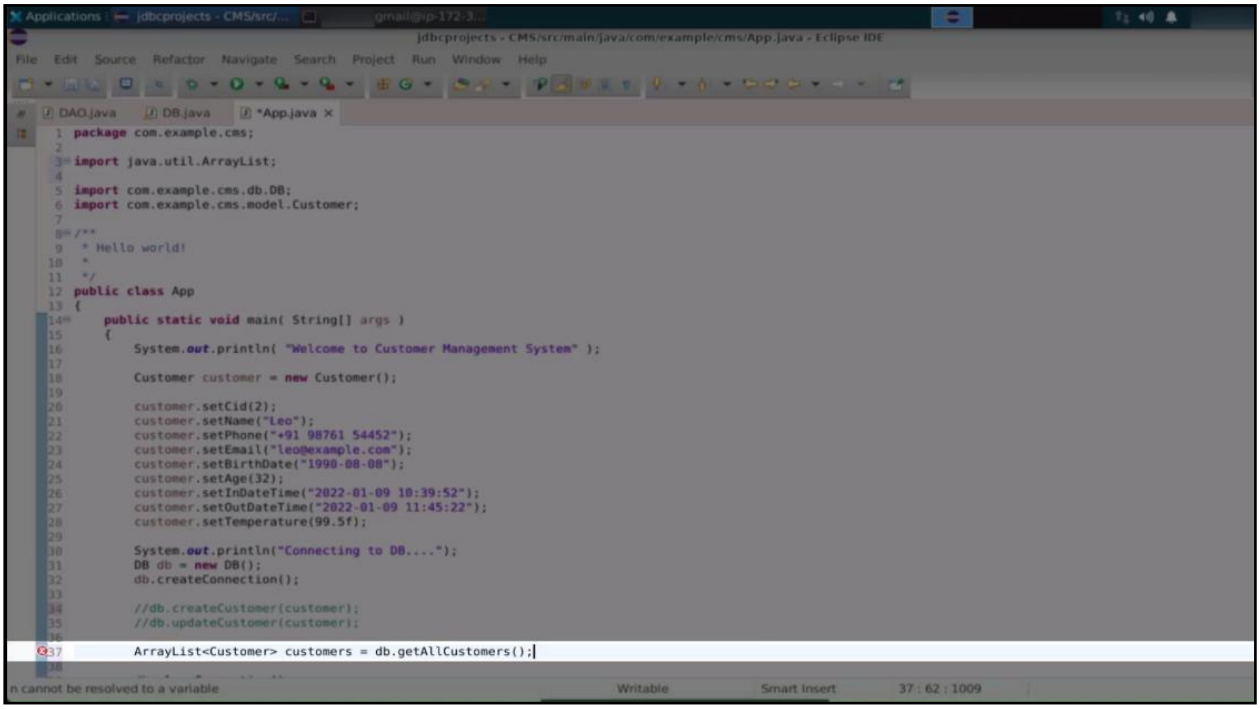
5.1 Use the **getAllCustomers()** method to fetch details of all the customers



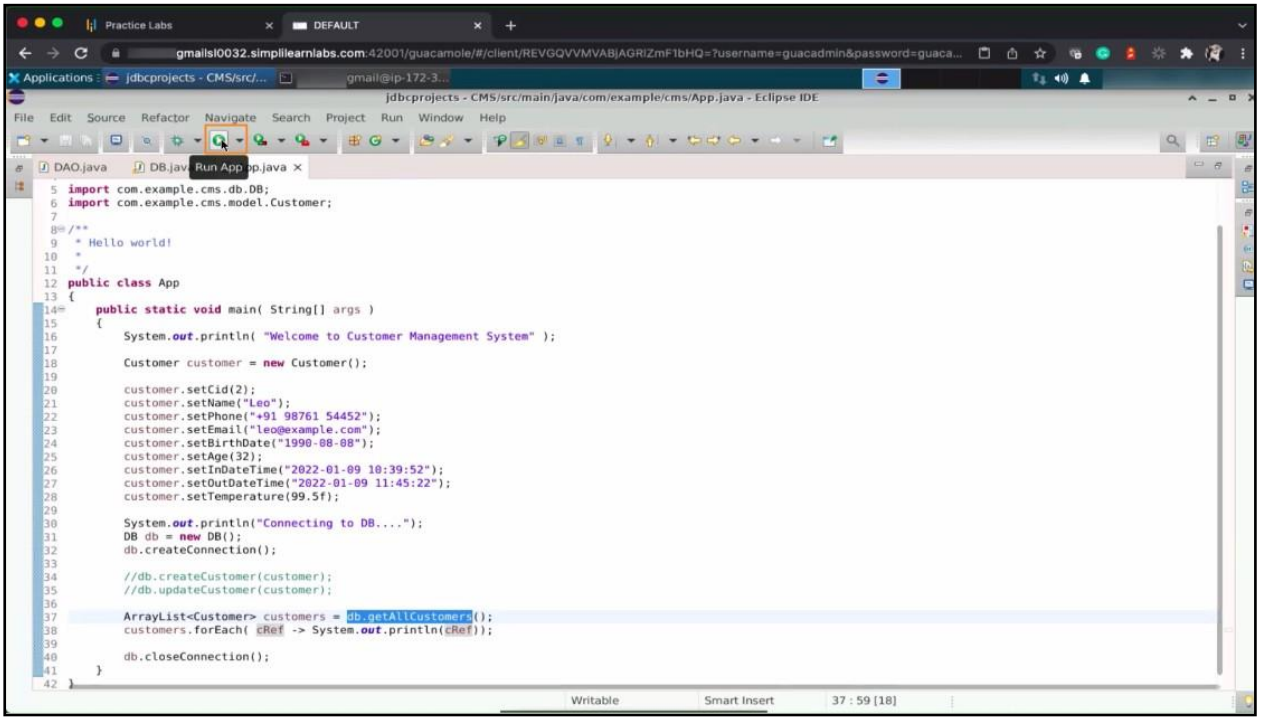
5.2 Enclose the select **statement** and connection string within a **try-catch block** to manage any potential exceptions



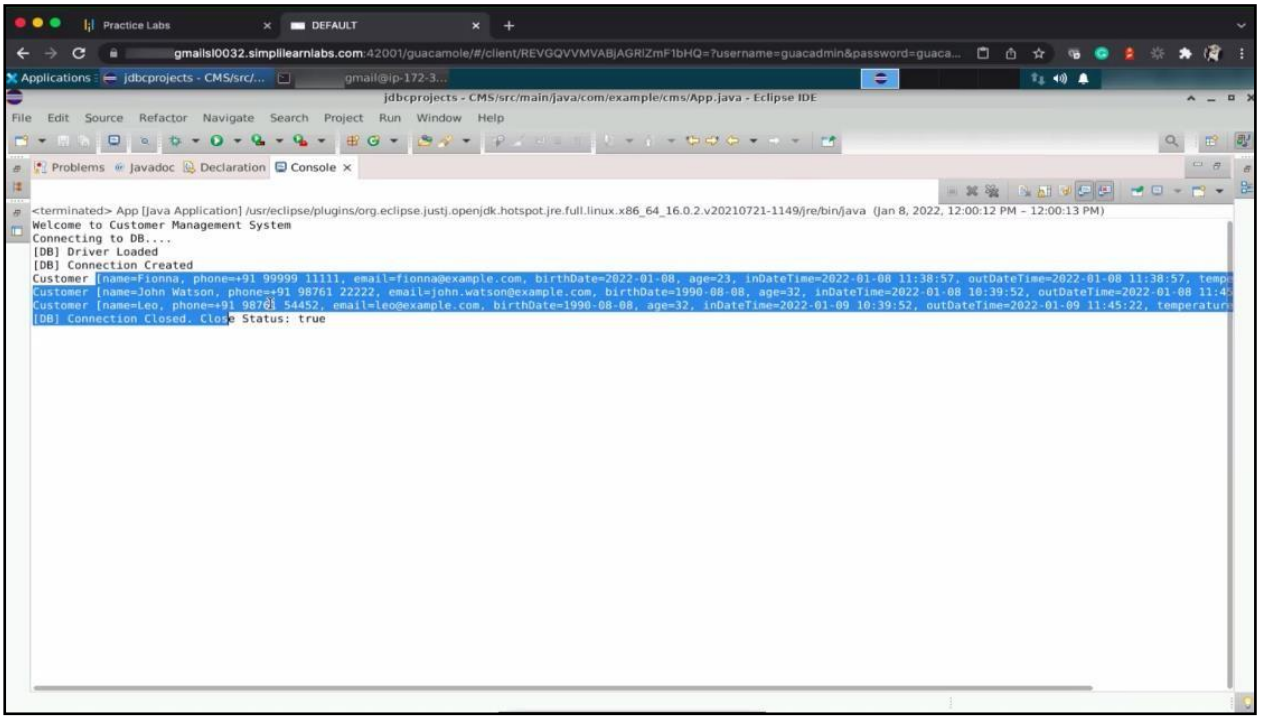
5.3 Return to the **App.java** file and call the **getAllCustomers()** method



5.4 Save and run the code

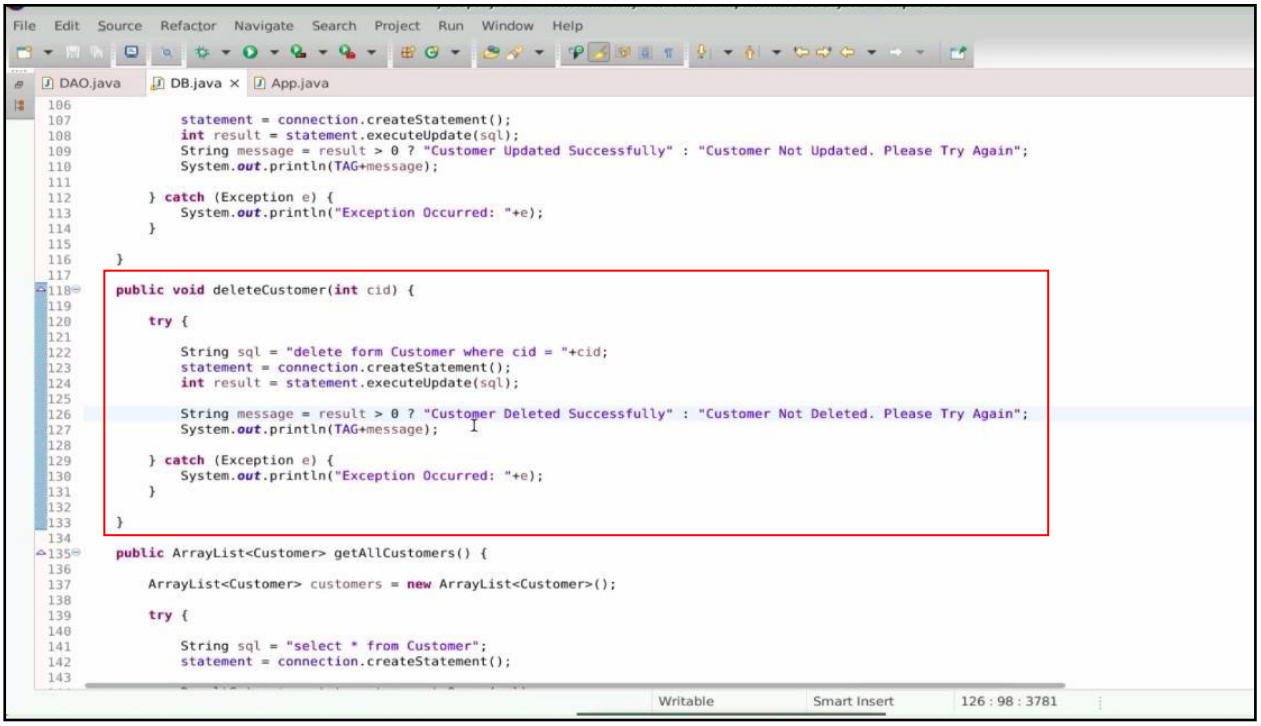


After running the code, the console will display the customer list as the output.

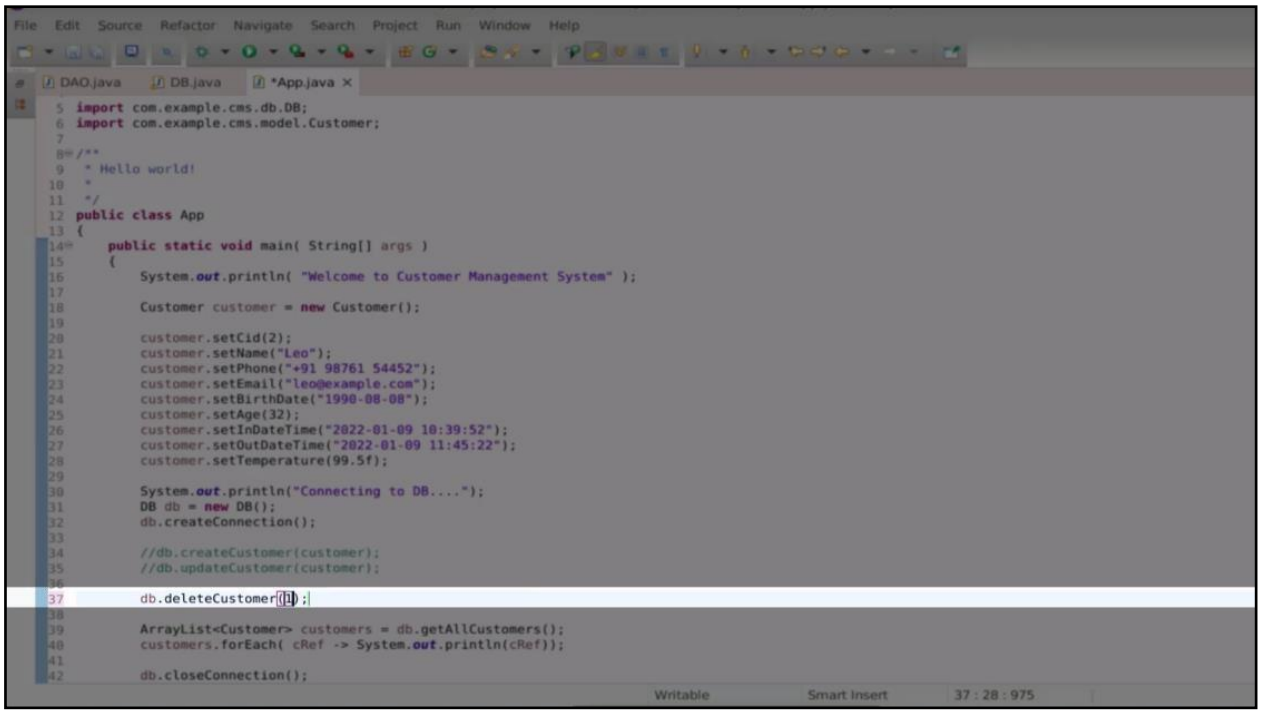


Step 6: Perform the delete operation

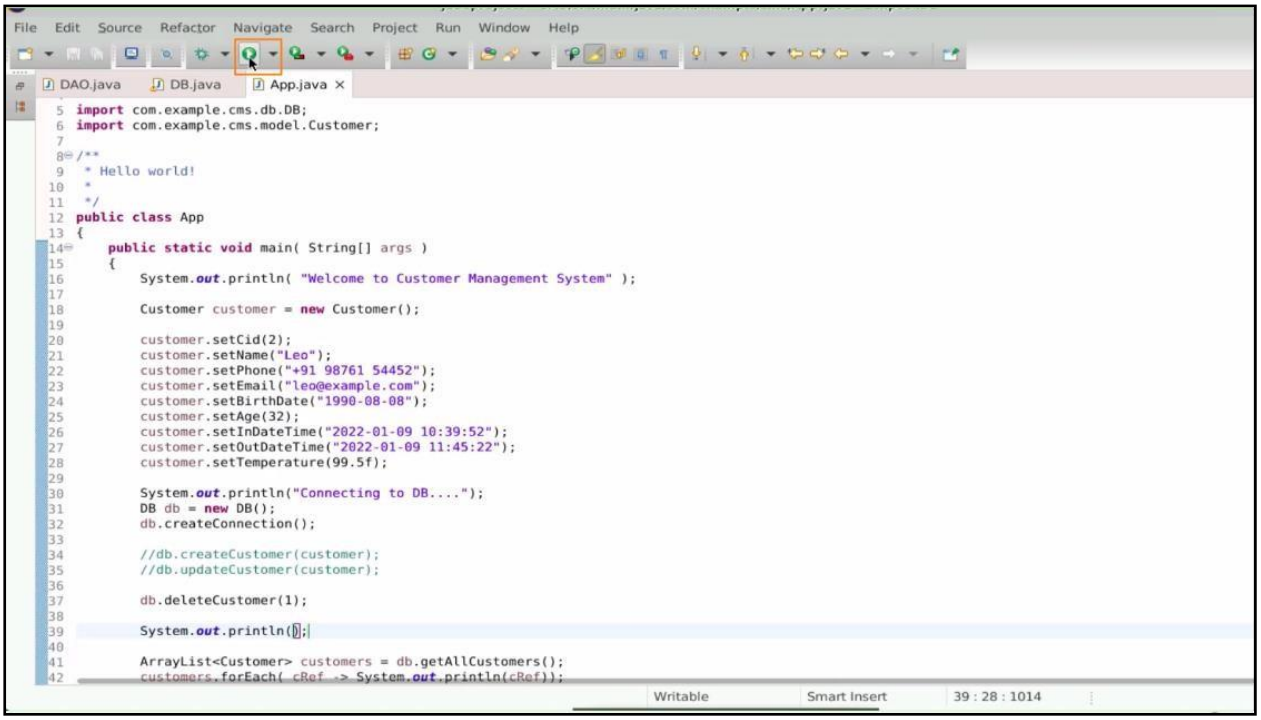
6.1 Type a delete operation inside the **try-catch** block, execute the **deleteCustomer()** method, and add a message to it



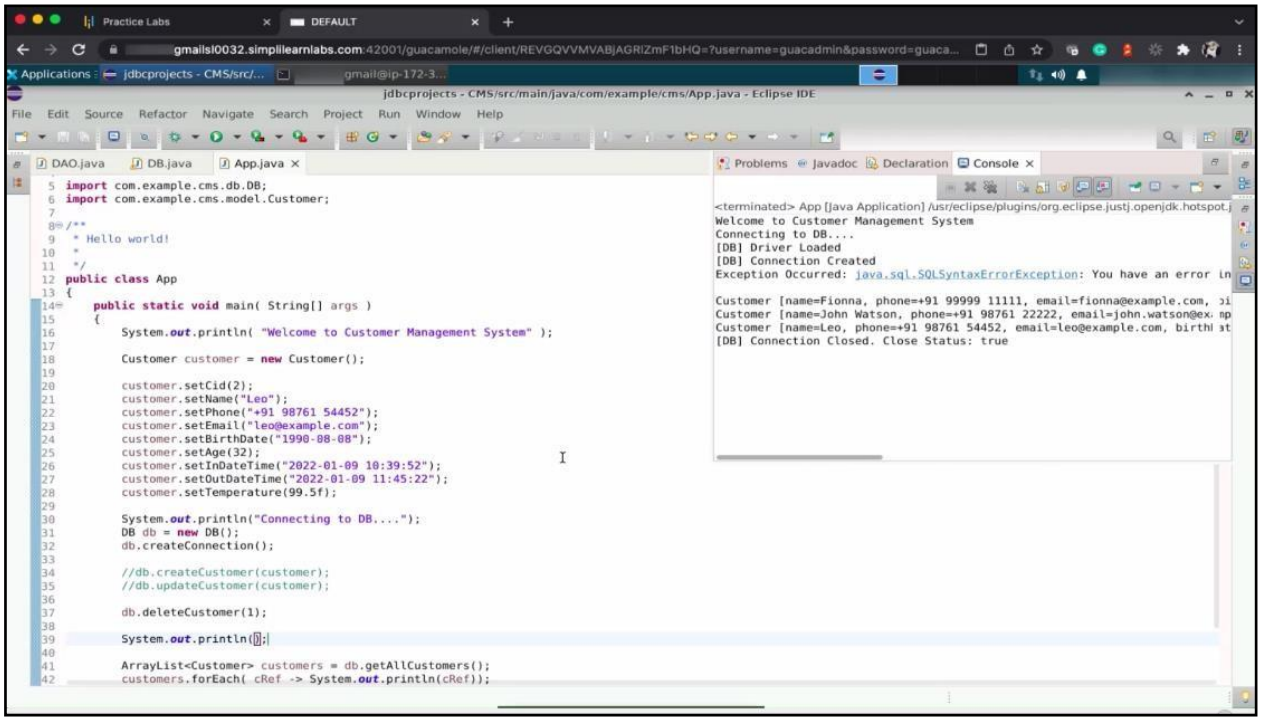
6.2 Return to the **App.java** file and write **db.deleteCustomer()** to call the function with the object **db**



6.3 Save and run the code

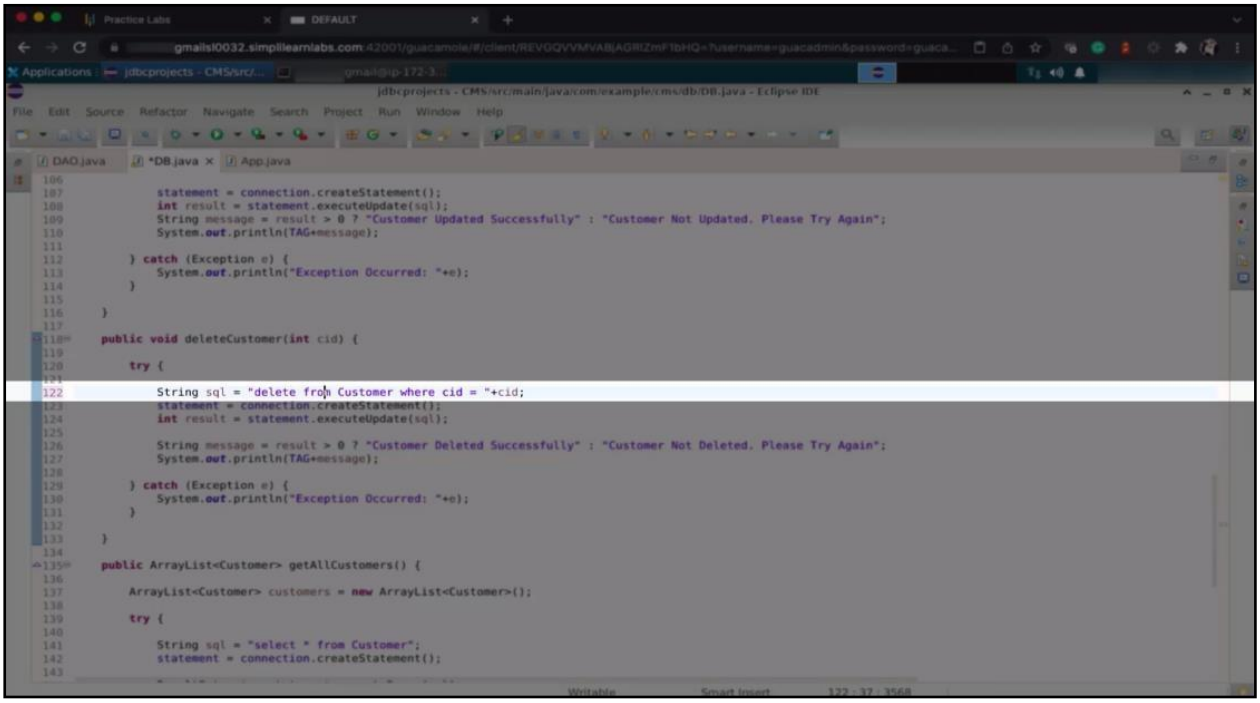


In the output, you will encounter an error indicated by the message **Exception occurred.**

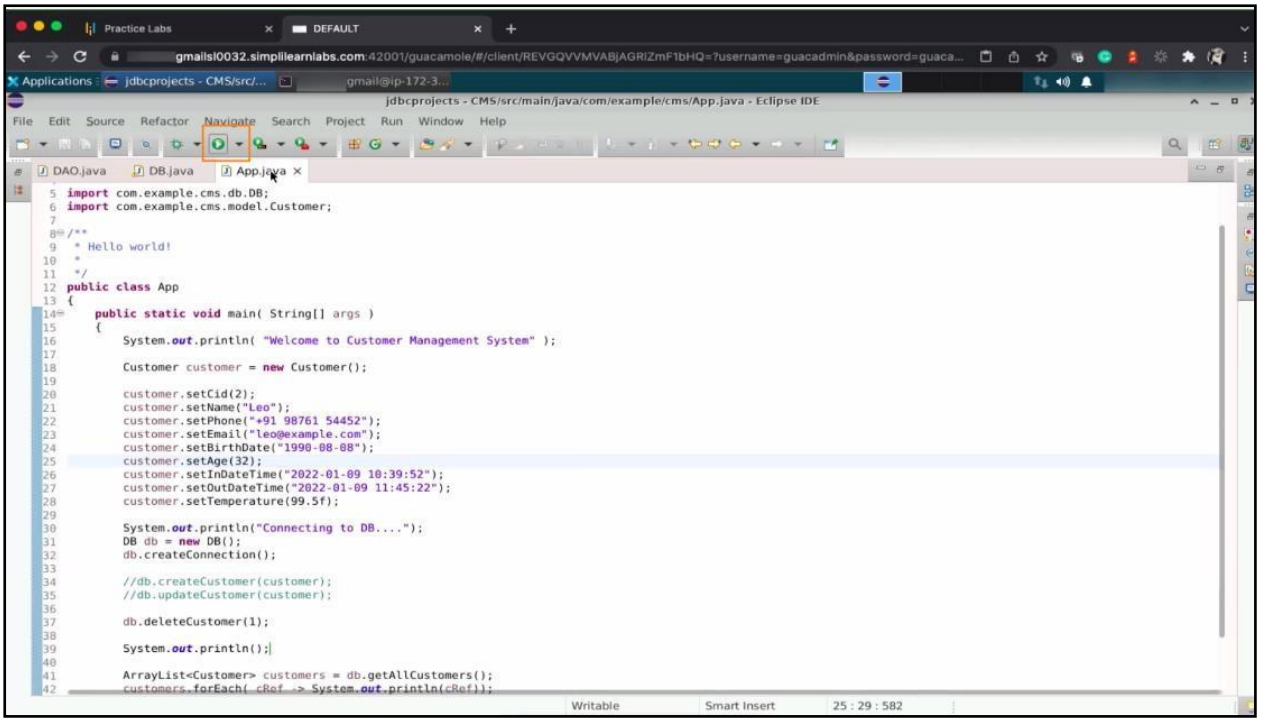


6.4 Update the following line of code to correct the delete query that caused the error:

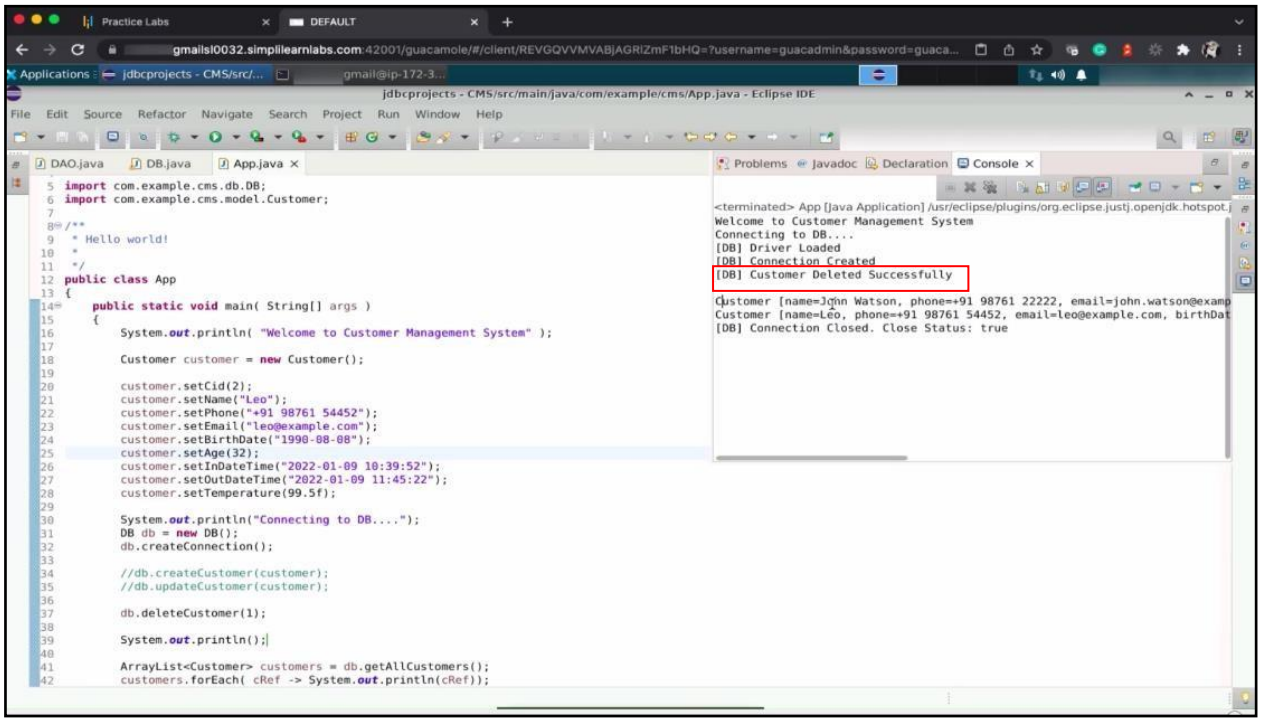
String sql= “delete from Customer where cid = “ +cid;



6.5 Return to the **App.java** file and run the code

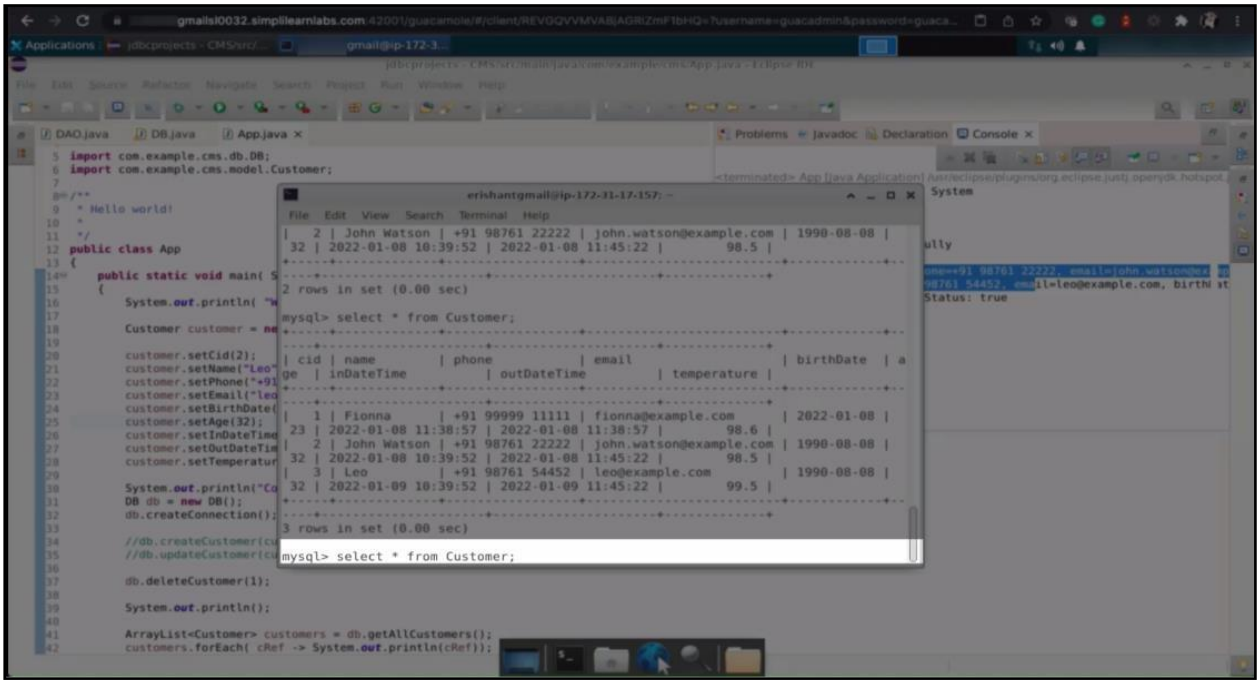


By following these steps, you will successfully deleted the customer, as indicated by the message **Customer Deleted Successfully**.

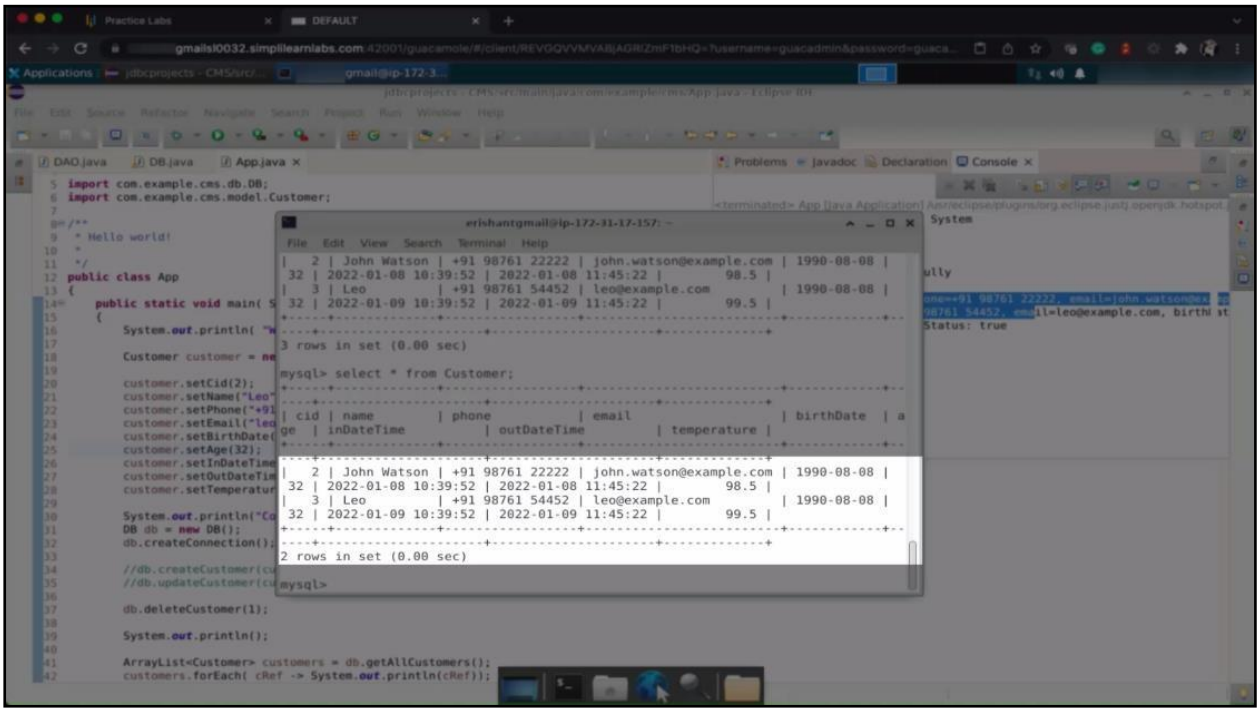


6.6 Return to the terminal and run the select command:

select * from Customers;



You will notice that the customer data has been deleted.



By following these steps, you have successfully demonstrated how CRUD (create, read, update, delete) operations can be performed in a Java project, reflecting changes in the database. We executed an insert query to add a new customer, an update query to modify existing customer details, the **getAllCustomers** operation to retrieve all customer records, and the **deleteCustomer()** function to remove a customer's record.