Lesson 01 Demo 02

Creating a Table in DB and Implementing ORM with OOPs

Objective: To create a customer model, a table in DB, and implement ORM with OOPs for managing customer data in a Java application

Tool required: Eclipse IDE

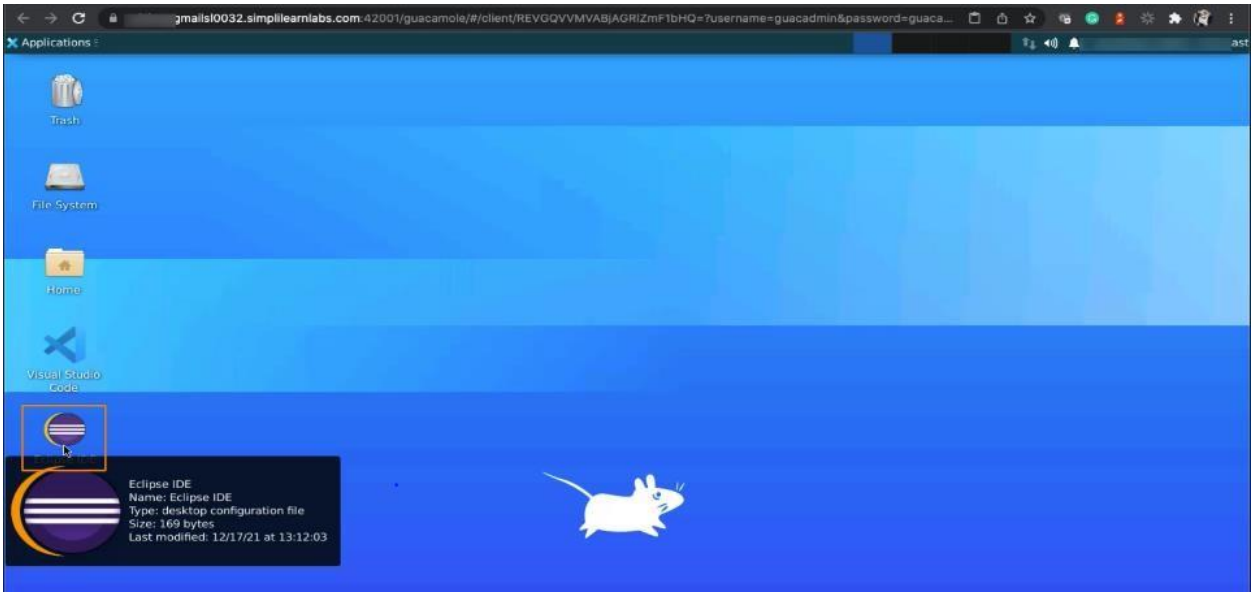
Prerequisites: Lesson 01 Demo 01

Steps to be followed:

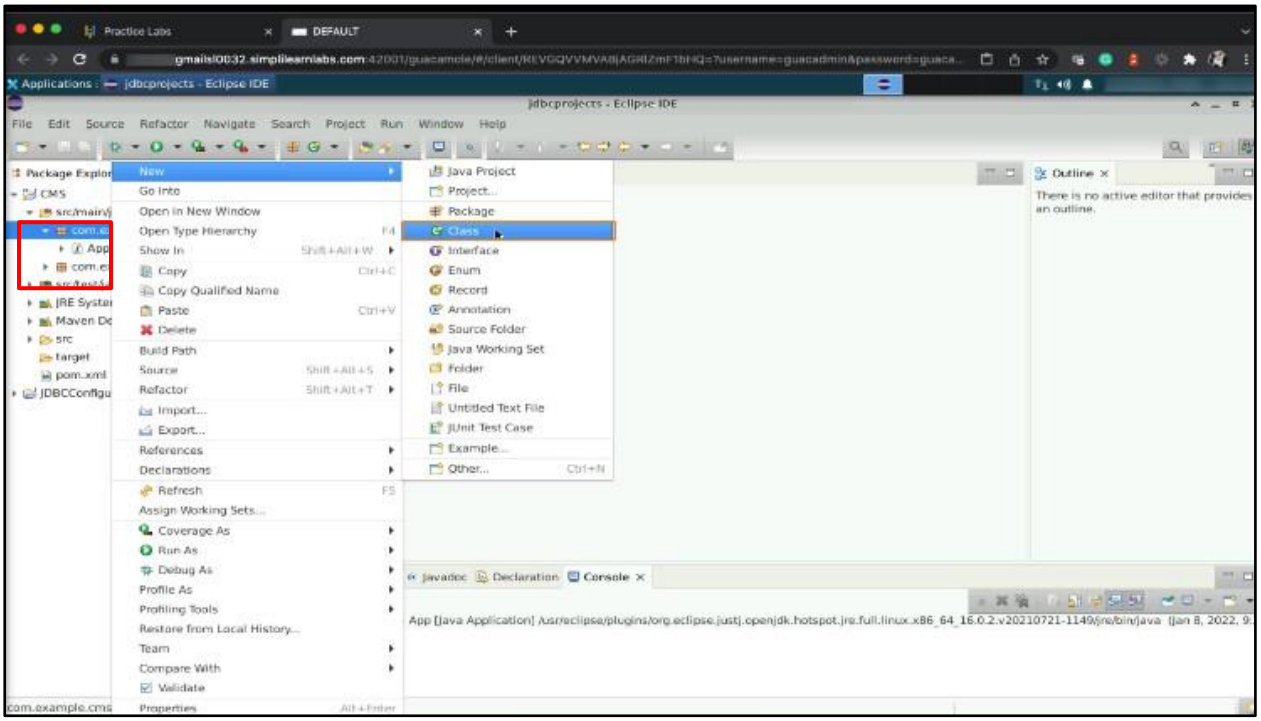
1. Create a new class
2. Write a customer object in App.java
3. Open the terminal window
4. Declare a method for a customer in the DAO interface

Step 1: Create a new class

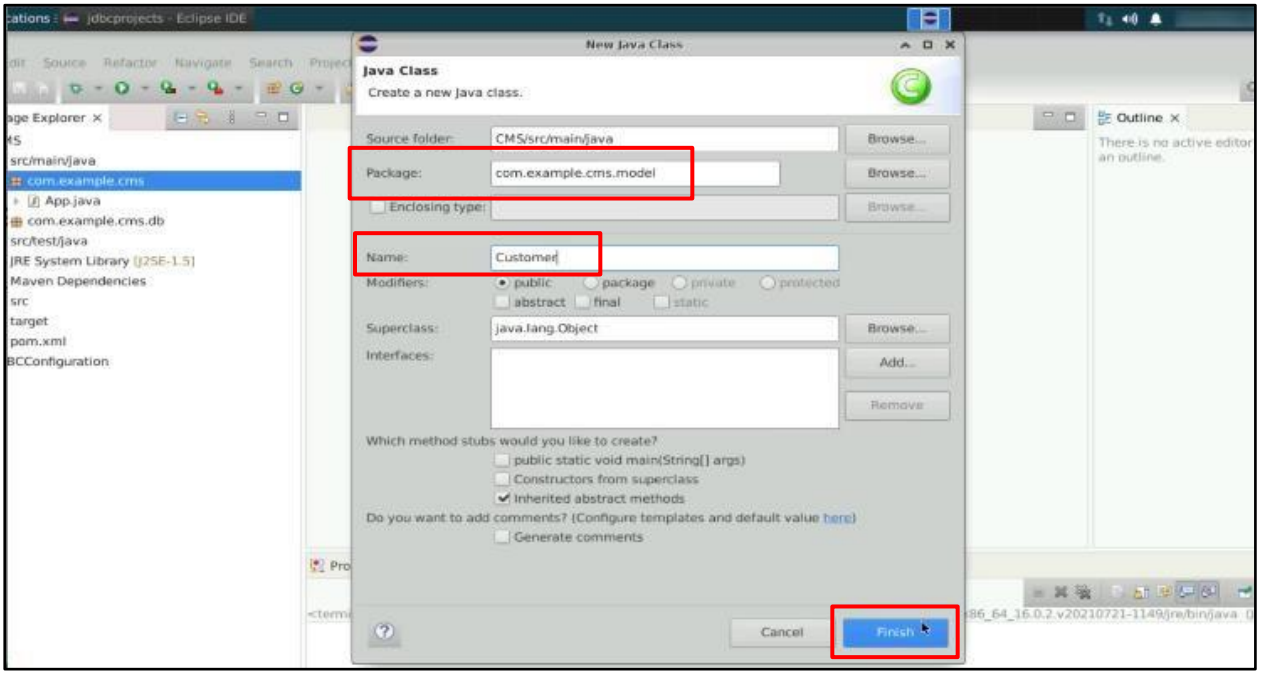
1.1 Open **Eclipse IDE** in your lab as shown in the screenshot below:



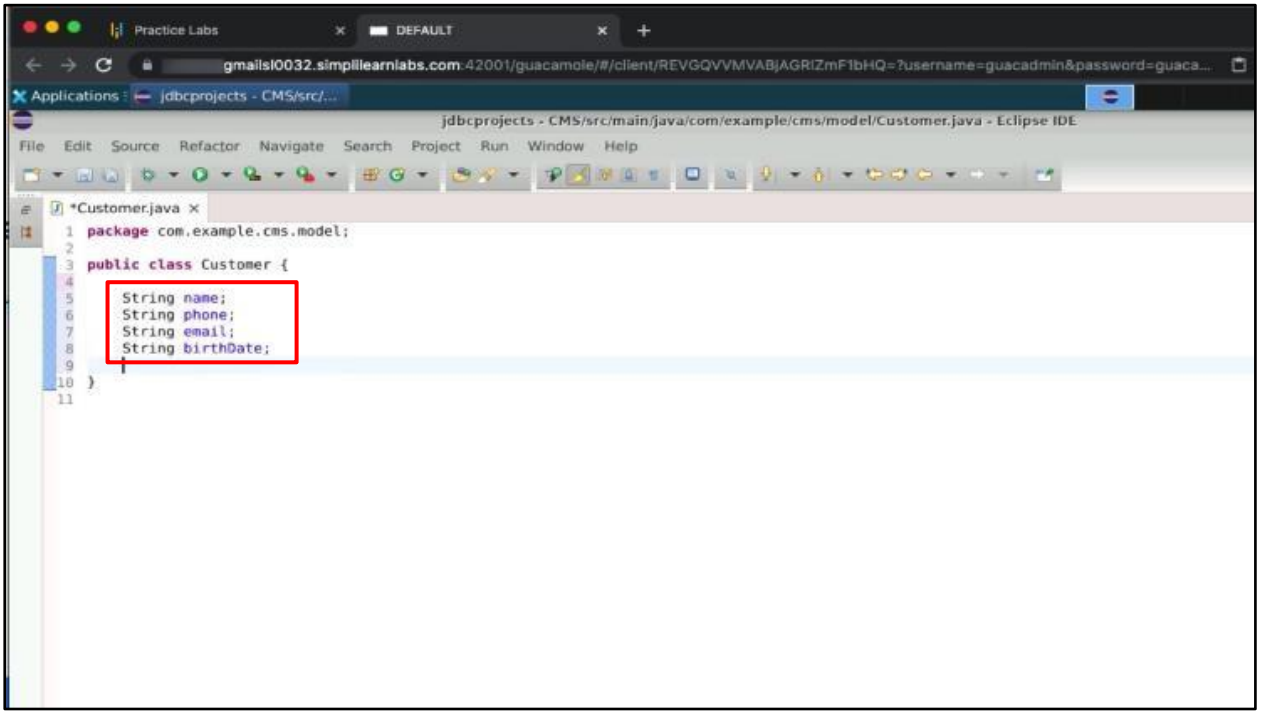
1.2 Right-click on **com.example.cms** in the **src** folder, select **New**, and click on **Class** to create a new class as shown in the screenshot below:



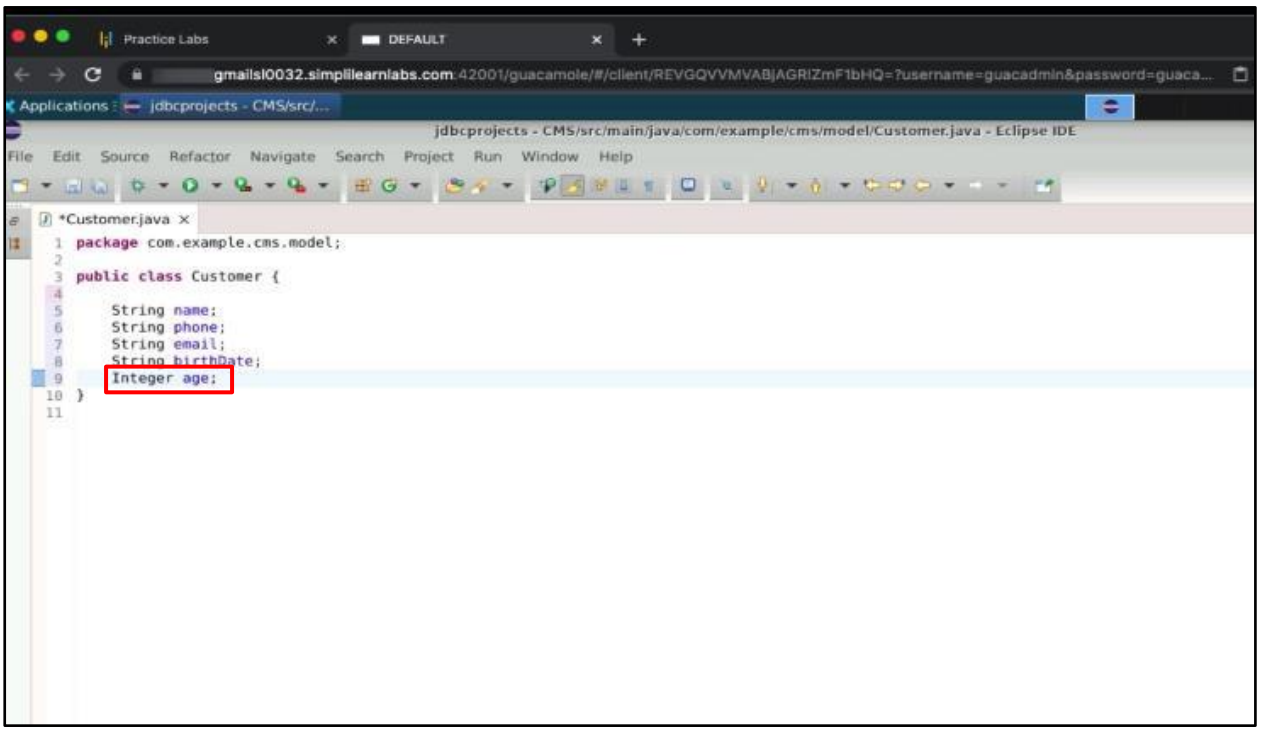
1.3 Add a name for class, package and click on **Finish** as shown in the screenshot below:



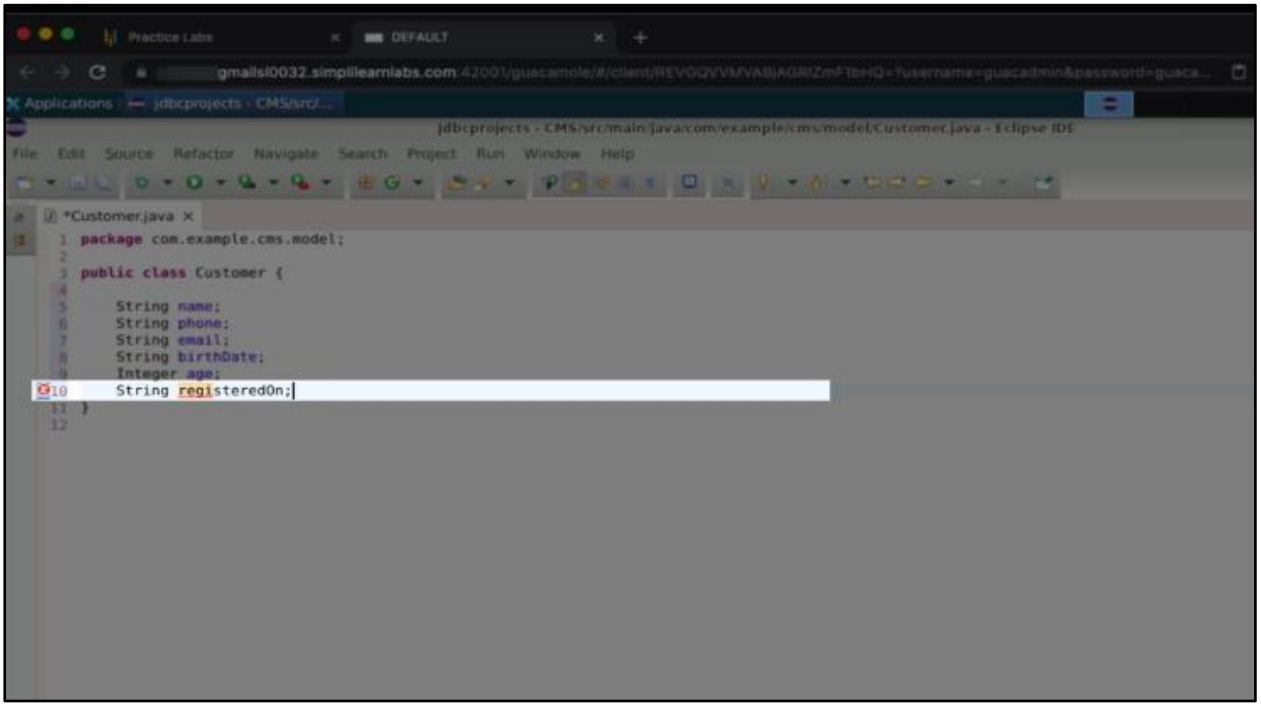
1.4 Add string attributes for the **Customer** database as shown in the below screenshot:



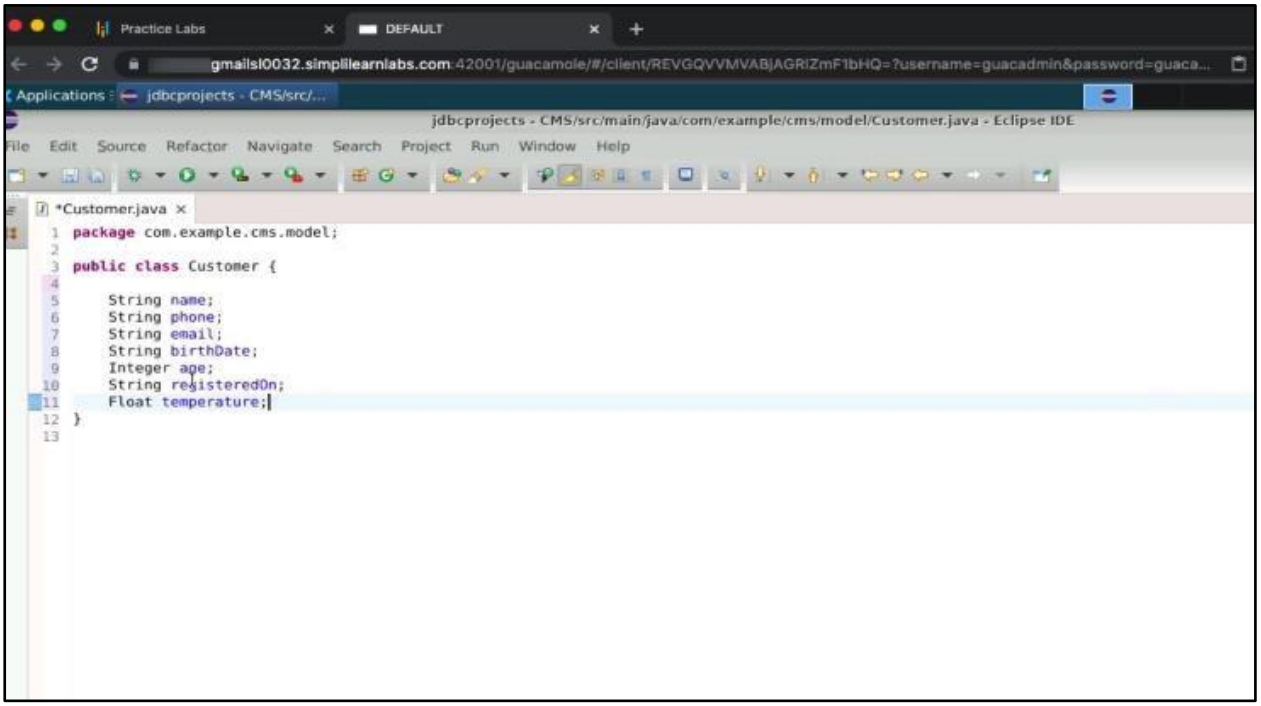
1.5 Add an integer attribute for **age** as shown in the below screenshot:



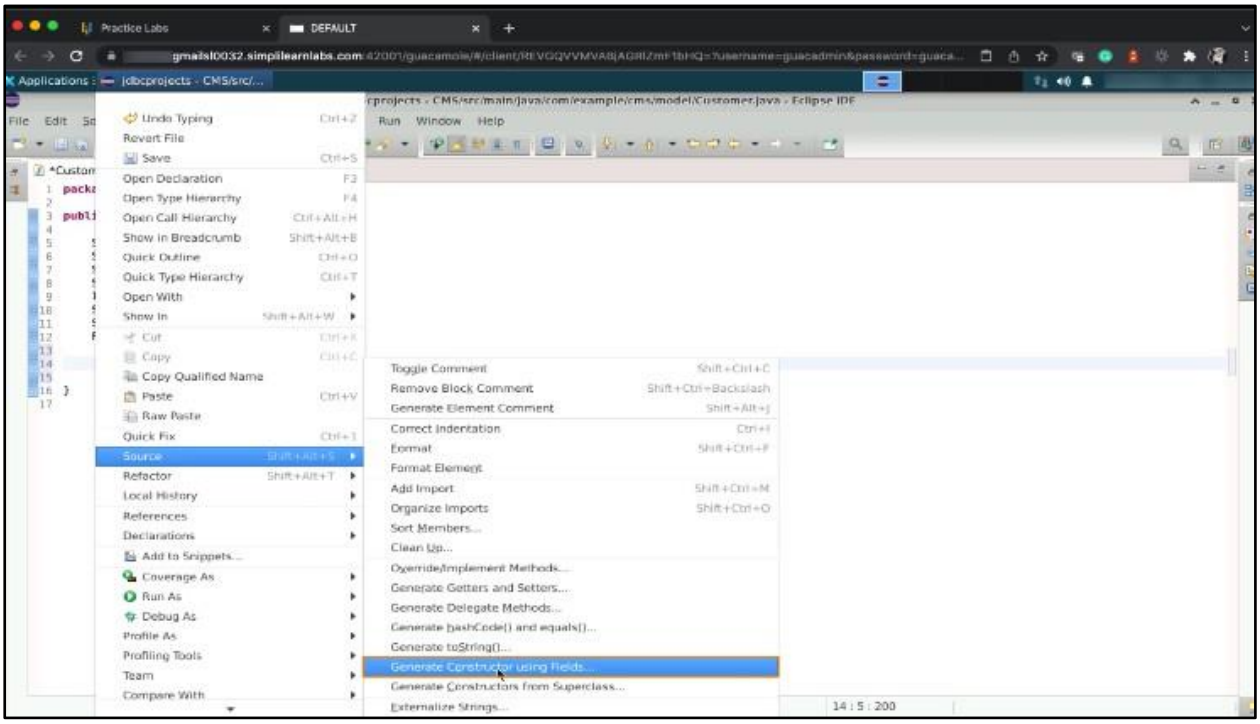
1.6 Add a string attribute for **registeredOn:**



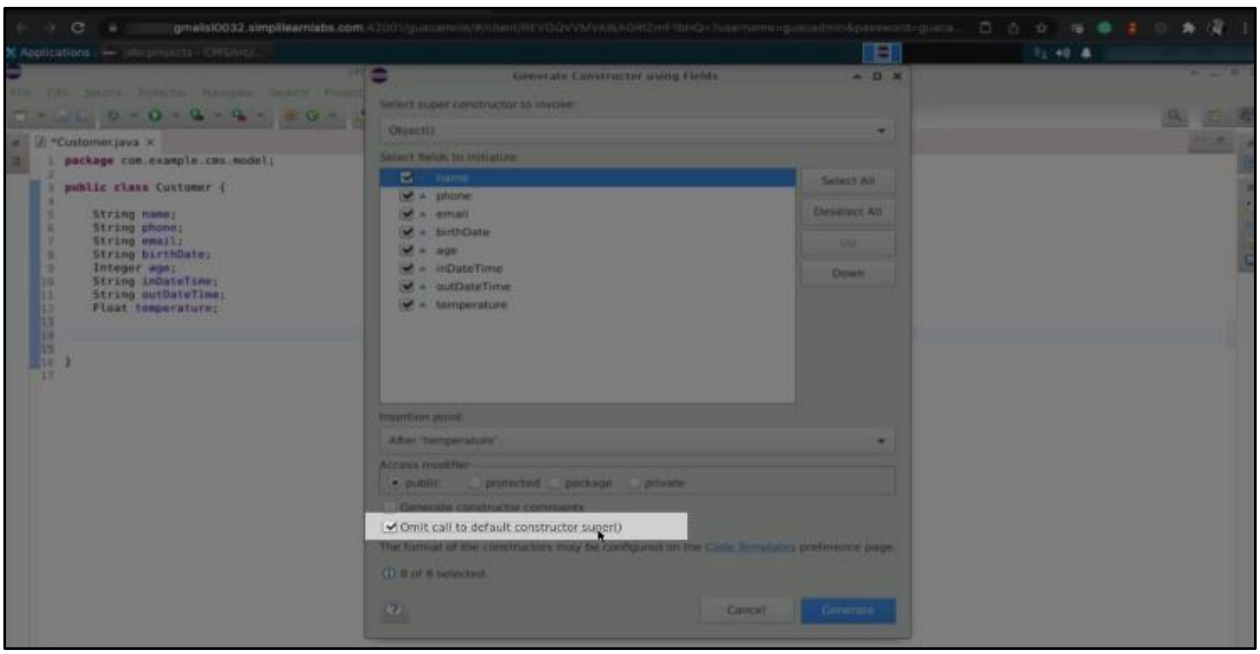
1.7 Add a floating-point attribute for **temperature**



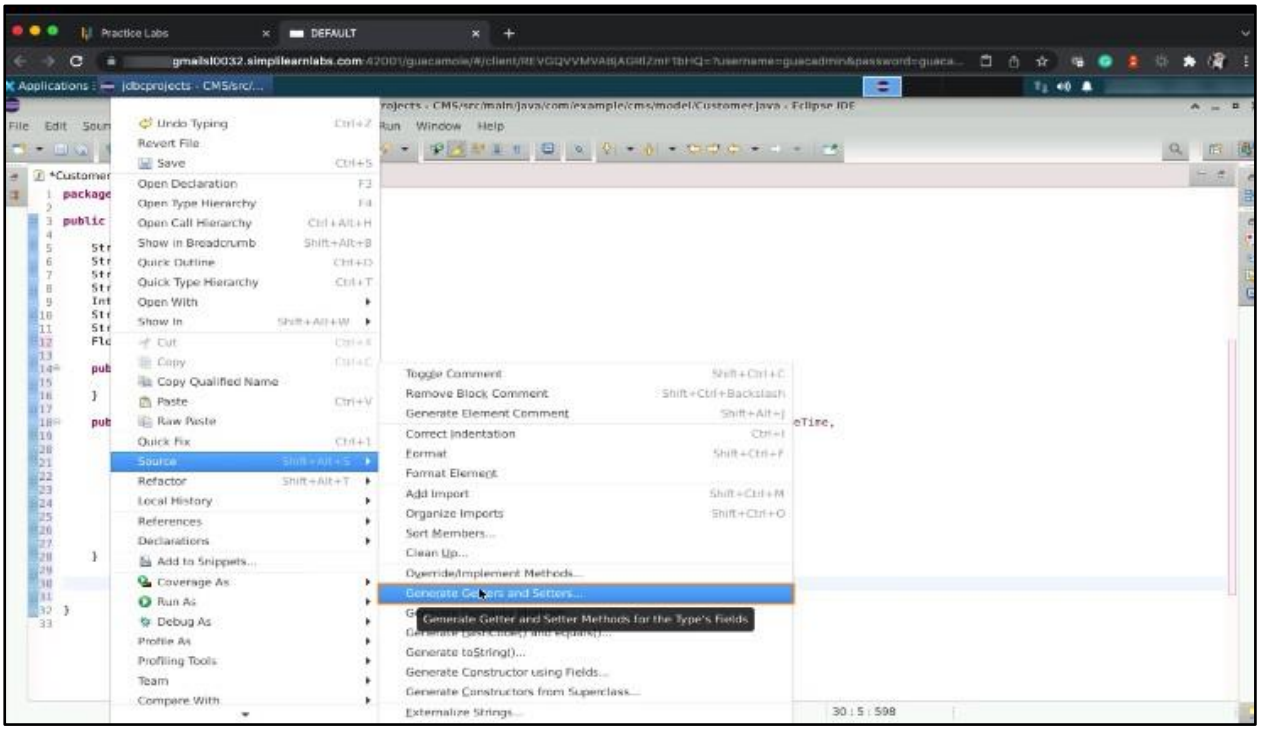
1.8 Right-click on class, select **Source,** and click on **Generate Constructor using Fields** to create a constructor



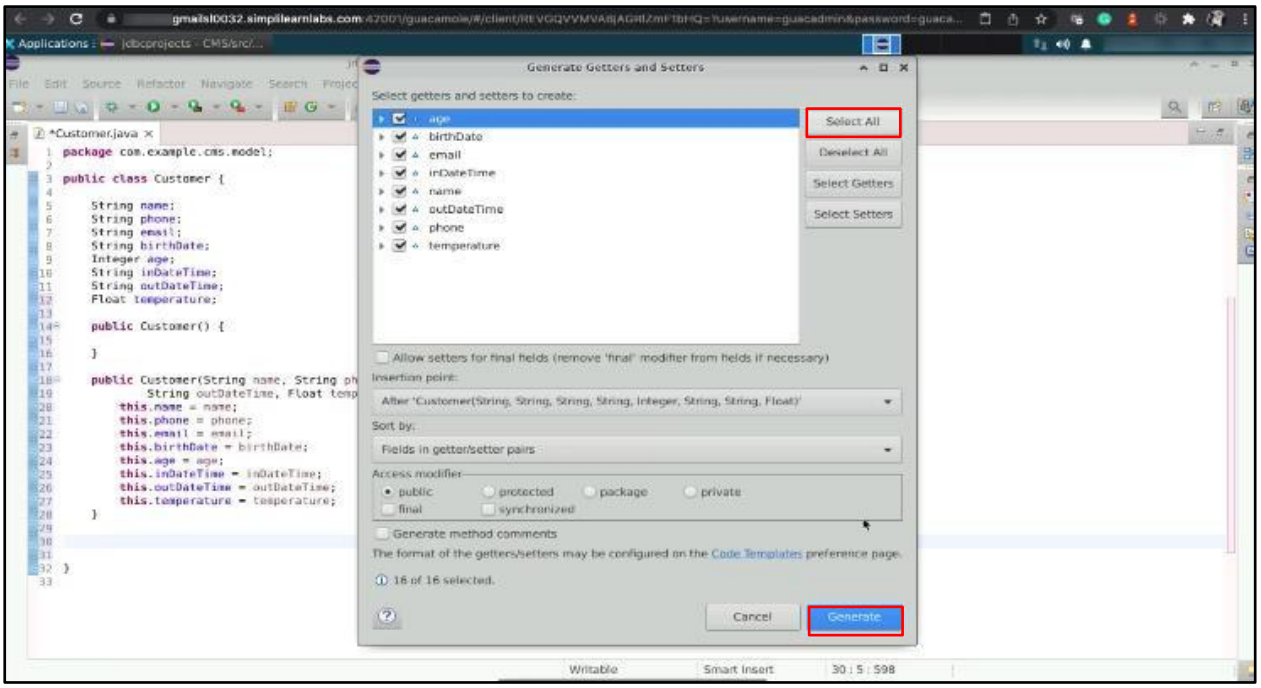
1.9 Mark the checkbox **Omit call to default constructor super()** and click on **Generate**. The constructor will be created.



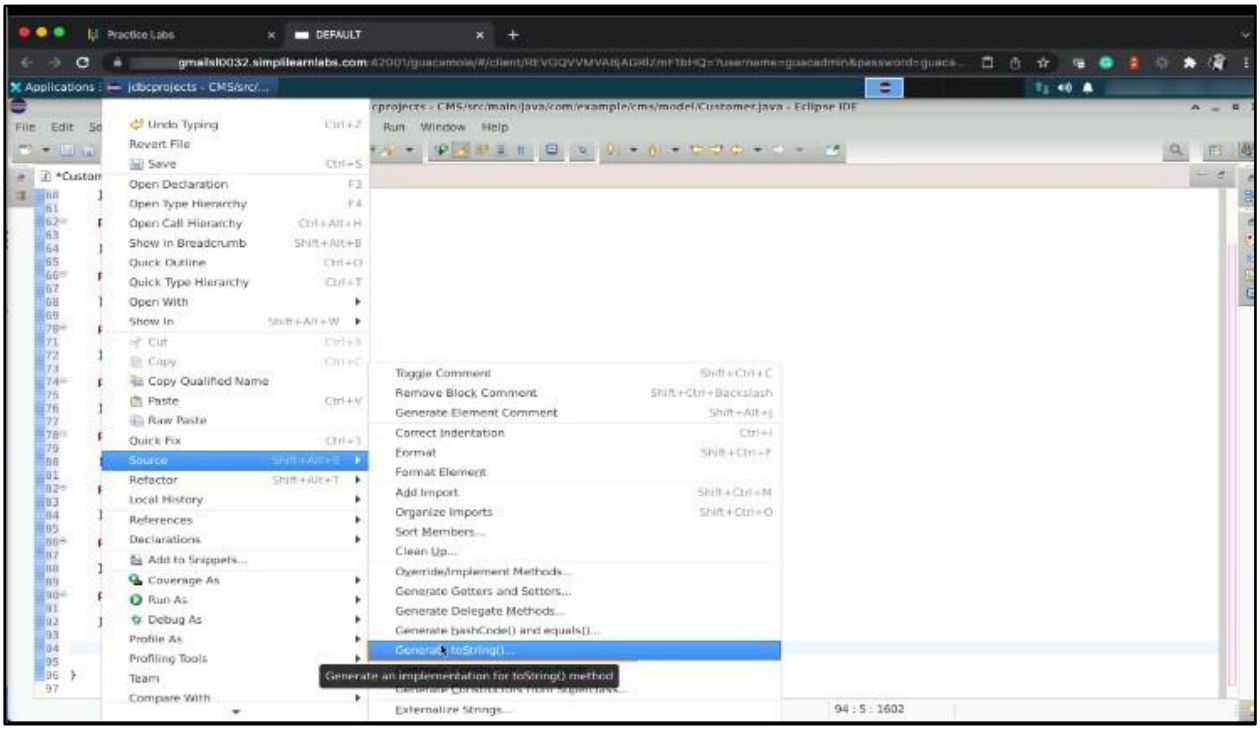
1.10 Right-click on class, select **Source,** and click on **Generate Getters and Setters** to write individual attributes



1.11 Click on **Select All** and then on **Generate**

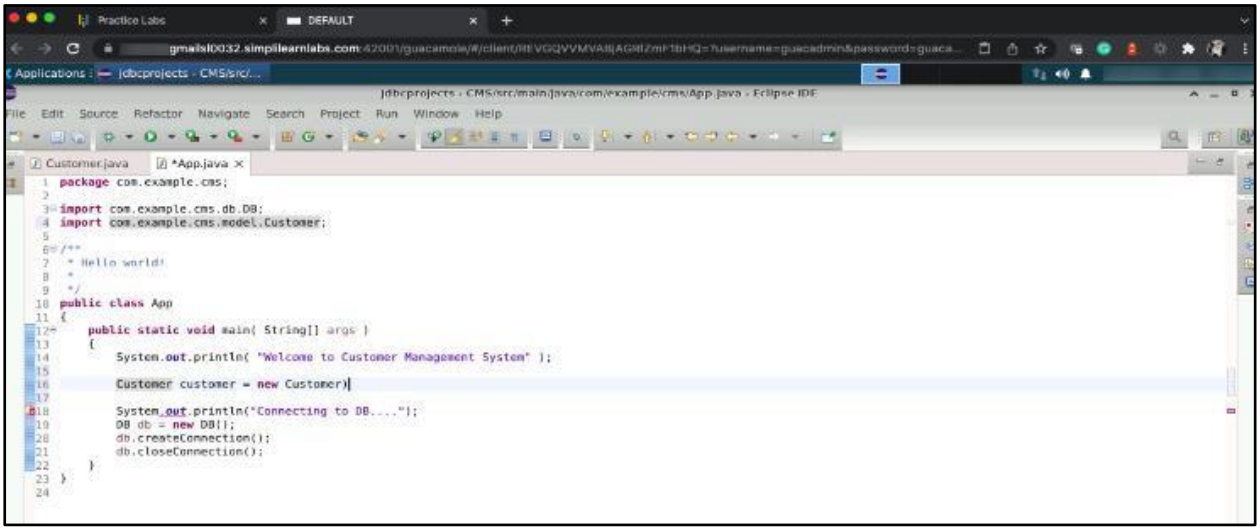


1.12 Right-click **Source** and select **Generate toString()** to decide the location of data in the object as shown in the screenshot below:

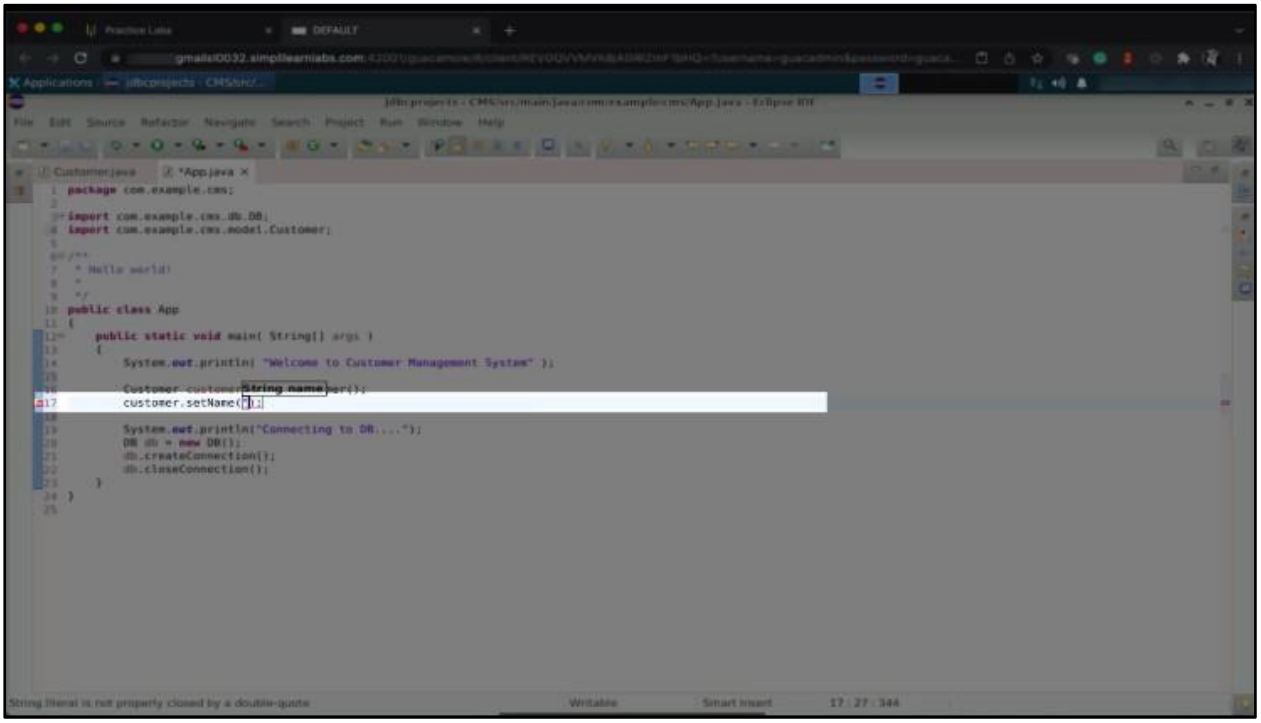


Step 2: Write a customer object in App.java

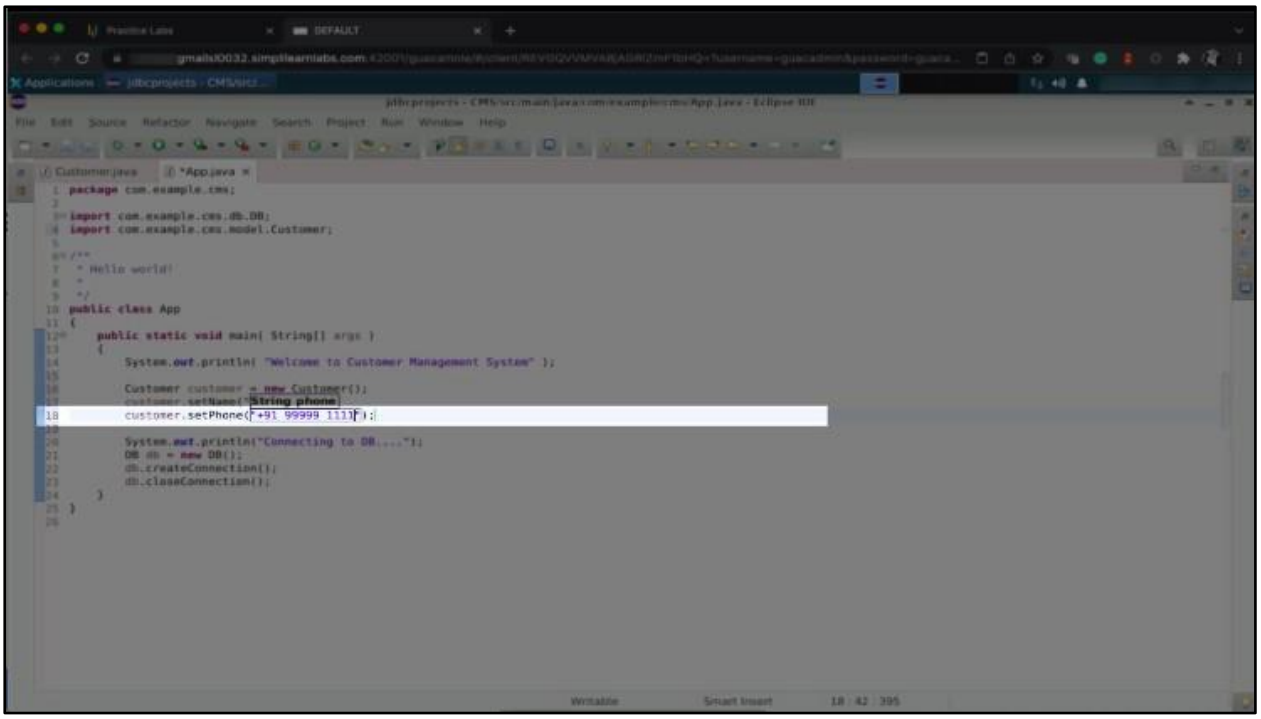
2.1 Define a **Customer** object in the **App.java** file



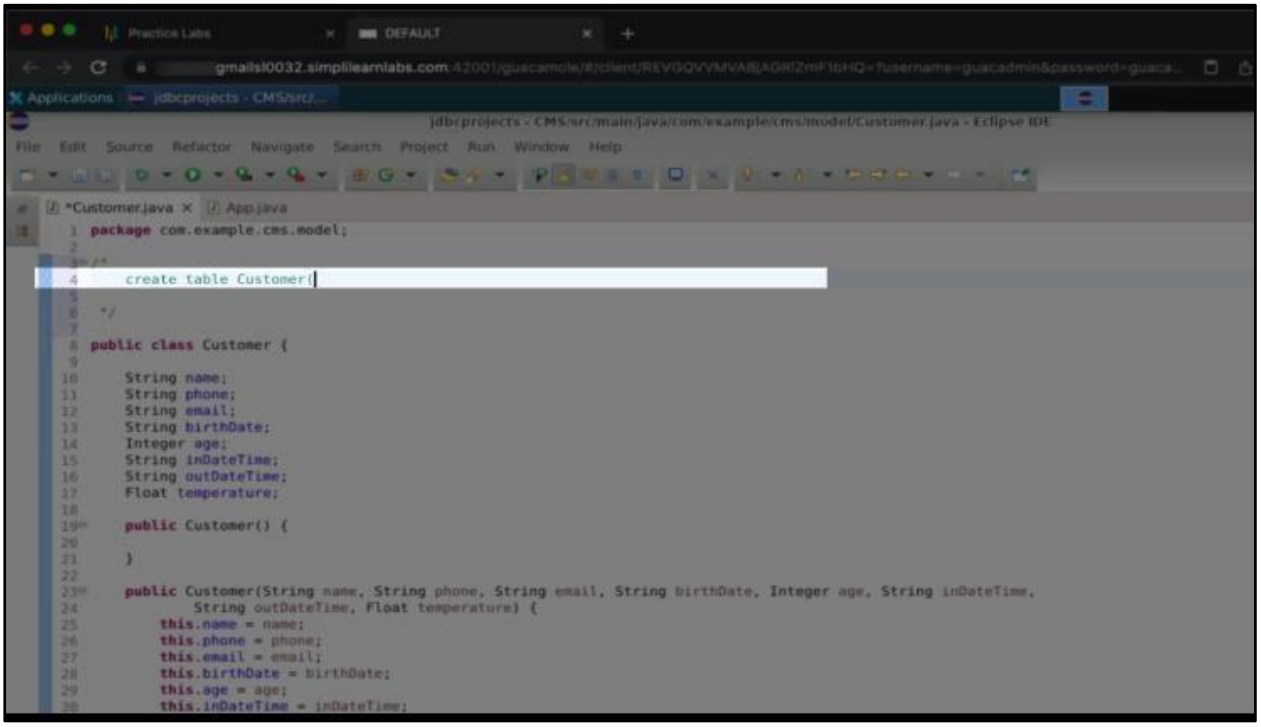
2.2 Set the name for the **Customer** object as shown in the screenshot below:



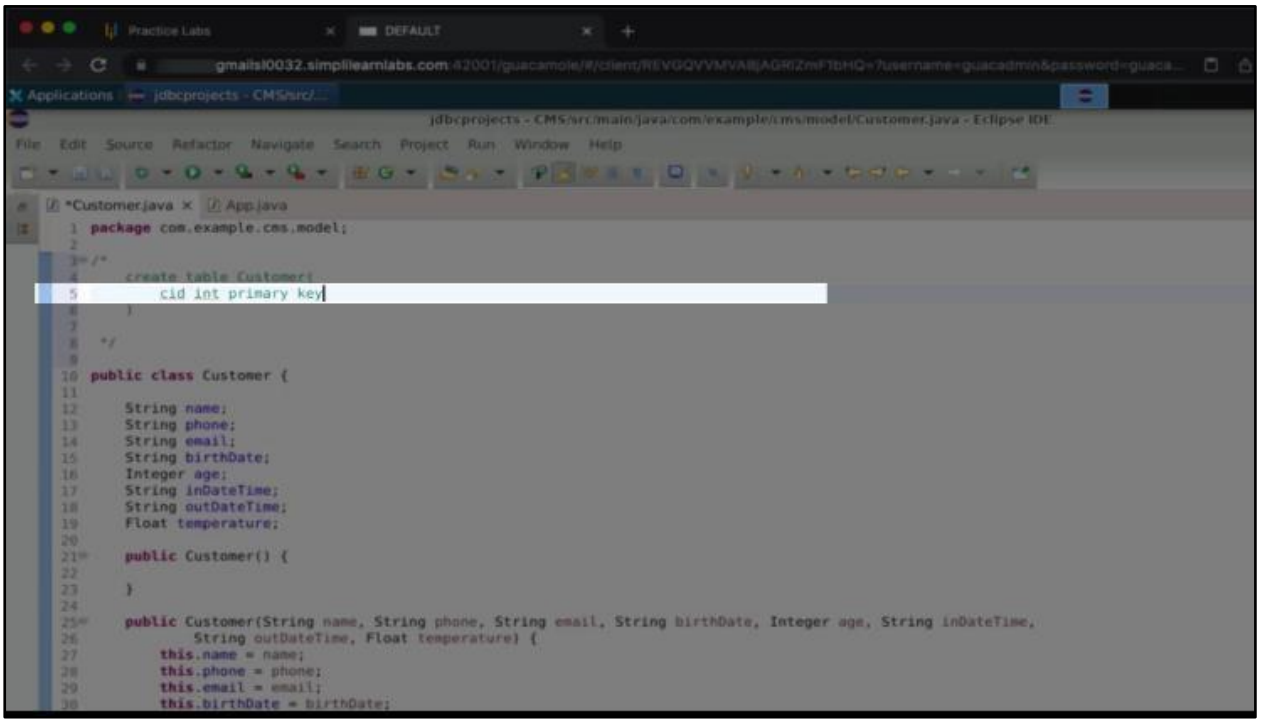
2.3 Create a method called **setPhone** with details as shown in the below screenshot:



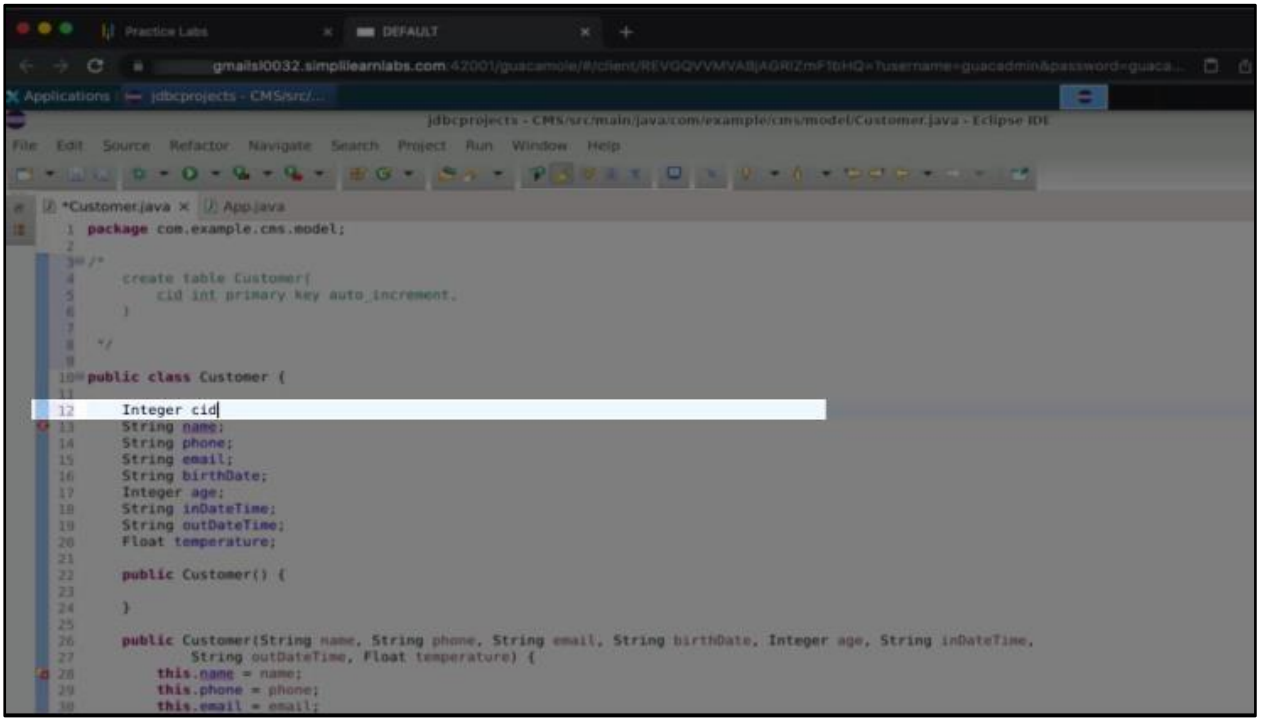
2.4 Use the **Customer** class and type **create table Customer** for object-relational mapping



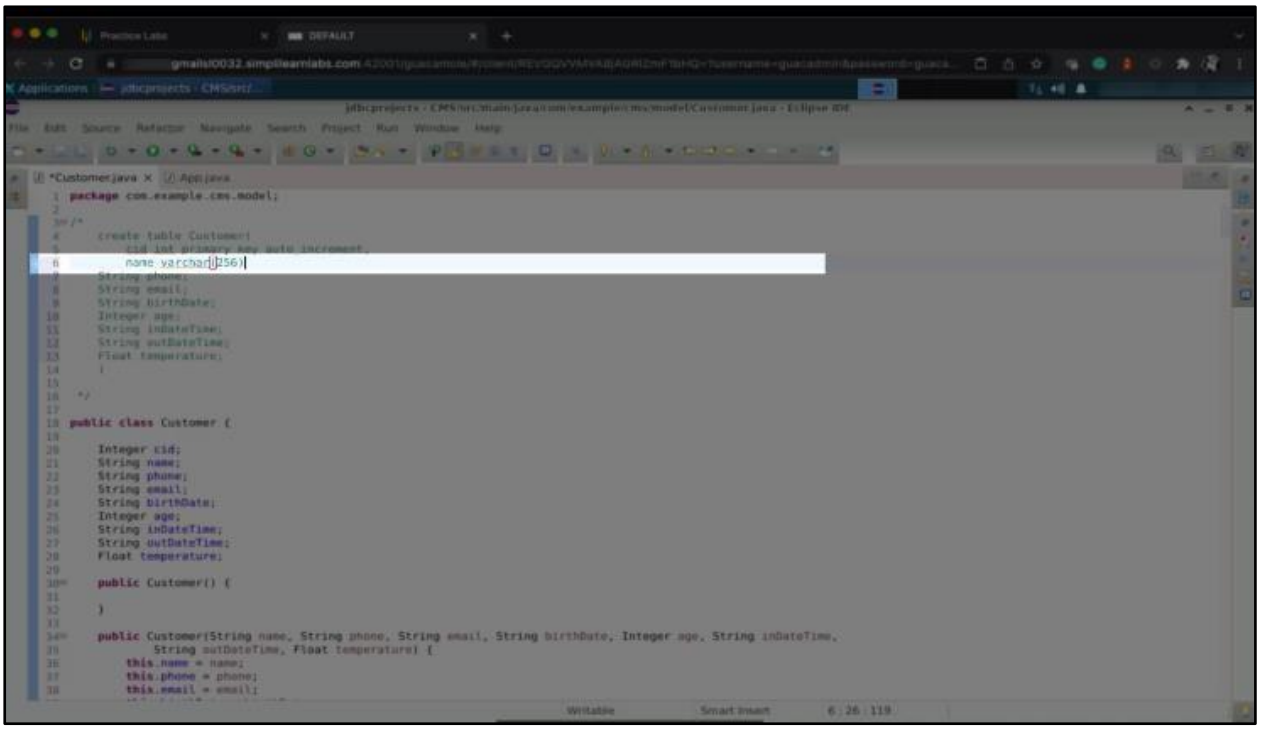
2.5 Add a primary key as **cid** for a unique identifier as shown below:



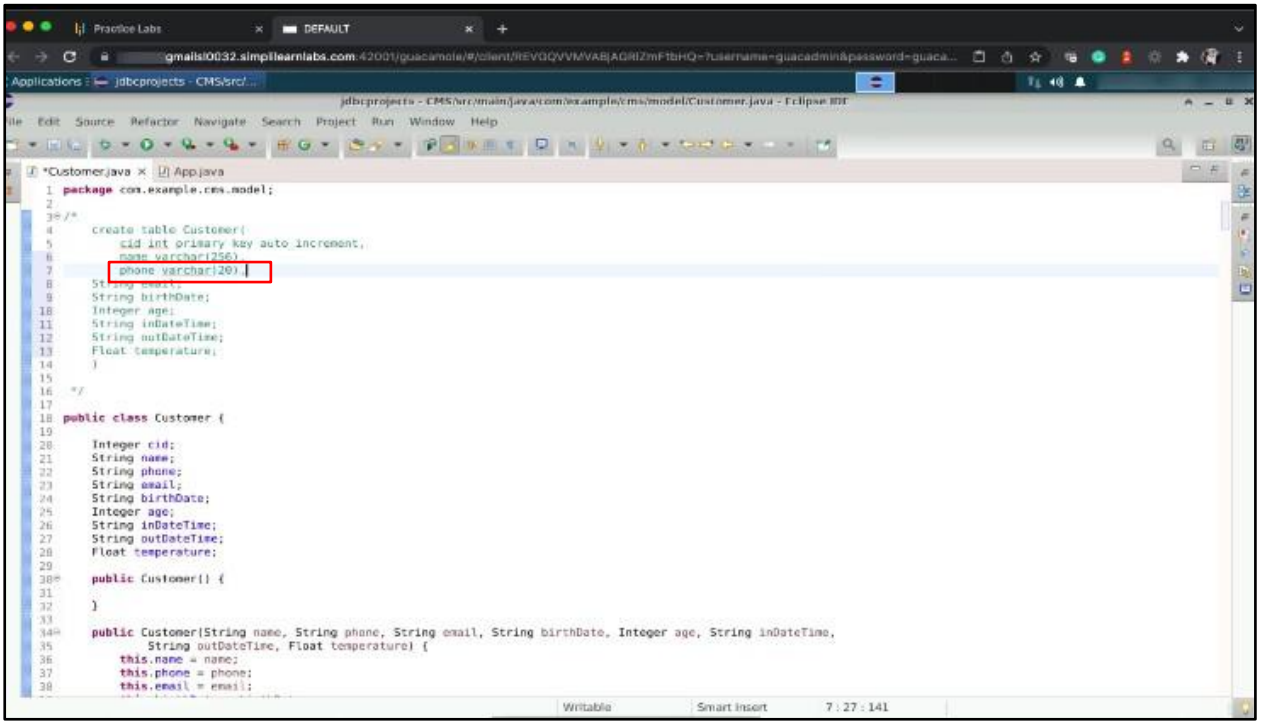
2.6 Add an integer attribute **cid** as shown in the below screenshot:



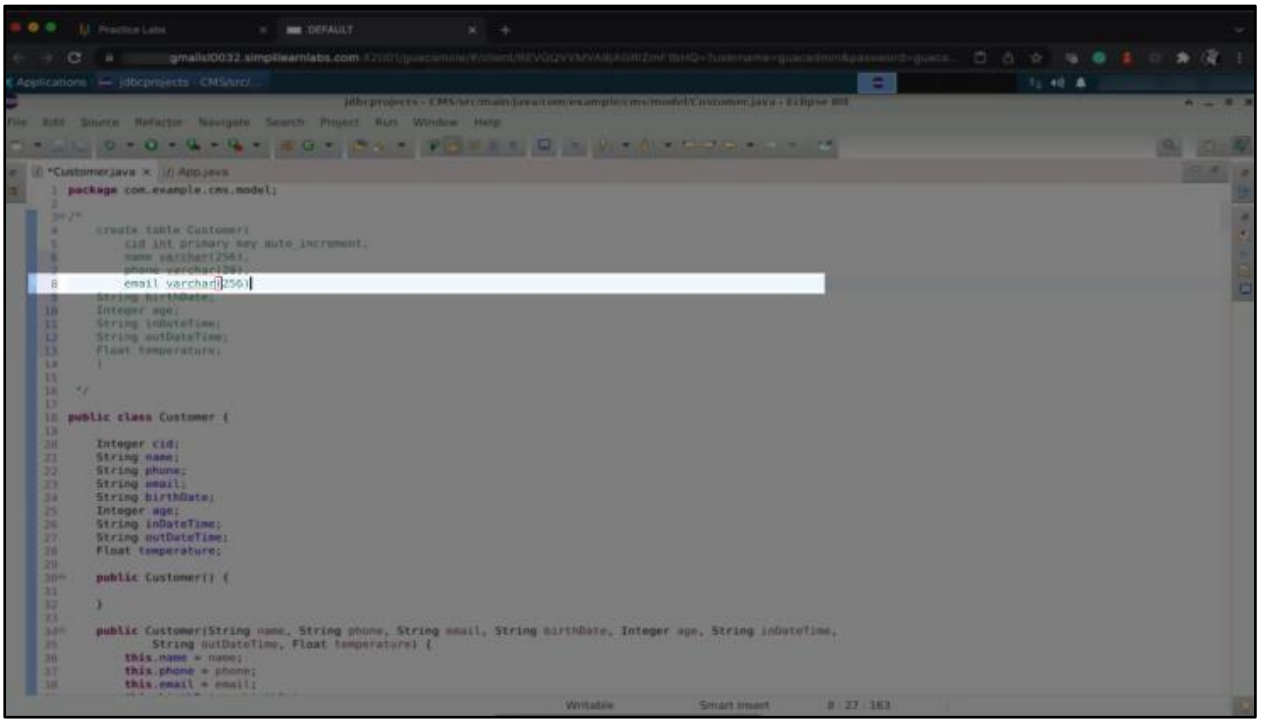
2.7 Specify the **name** attribute with the datatype **varchar(256)** as shown in the below screenshot:



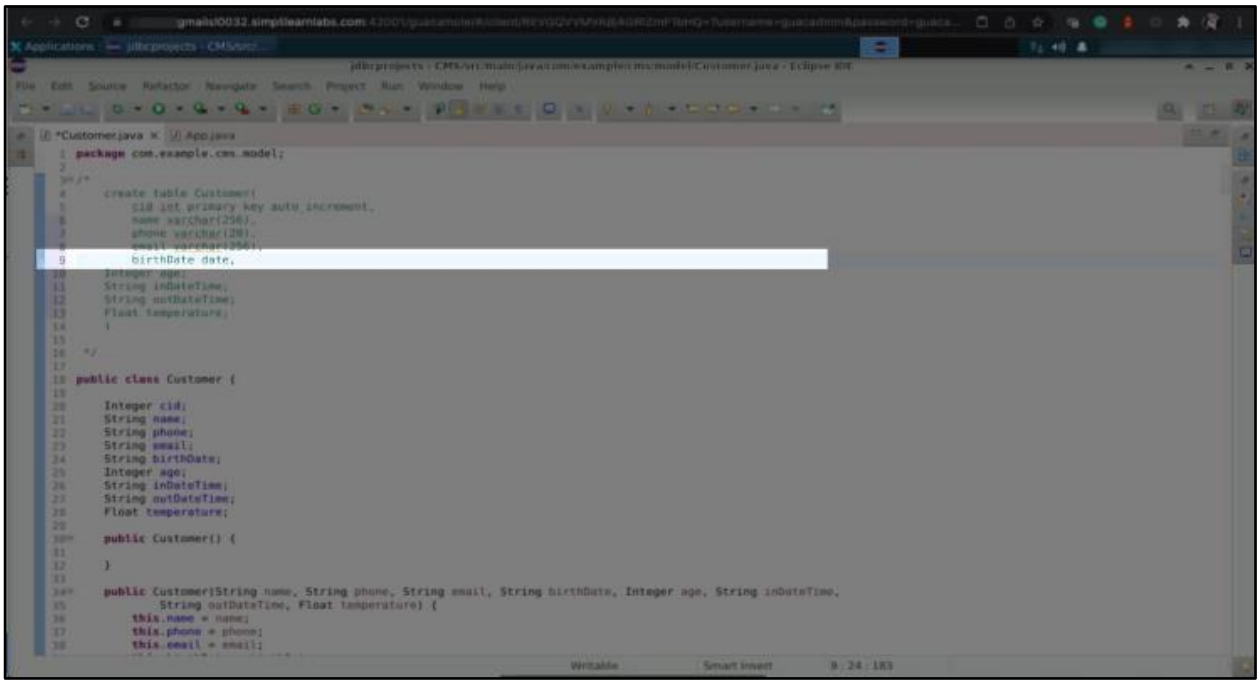
2.8 Add the **phone** attribute with the datatype **varchar(20)** as shown in the below screenshot:



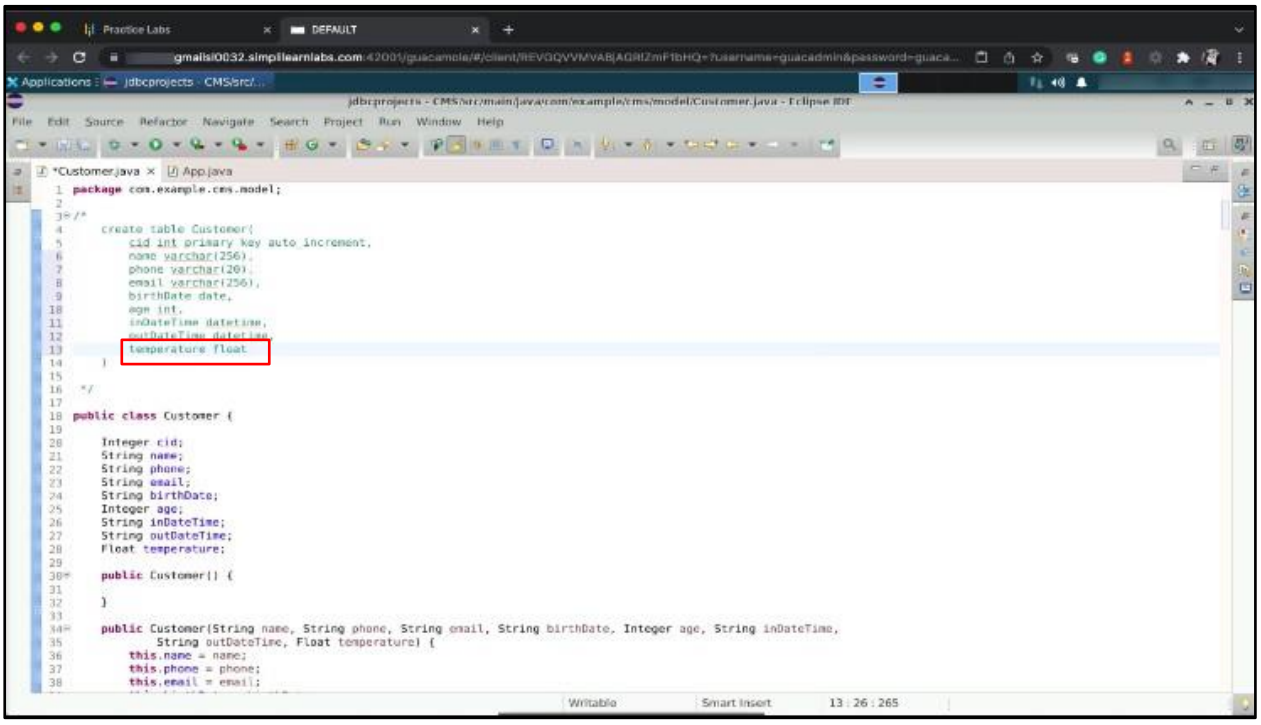
2.9 Add the **email** attribute with the datatype **varchar(256)** as shown in the below screenshot:



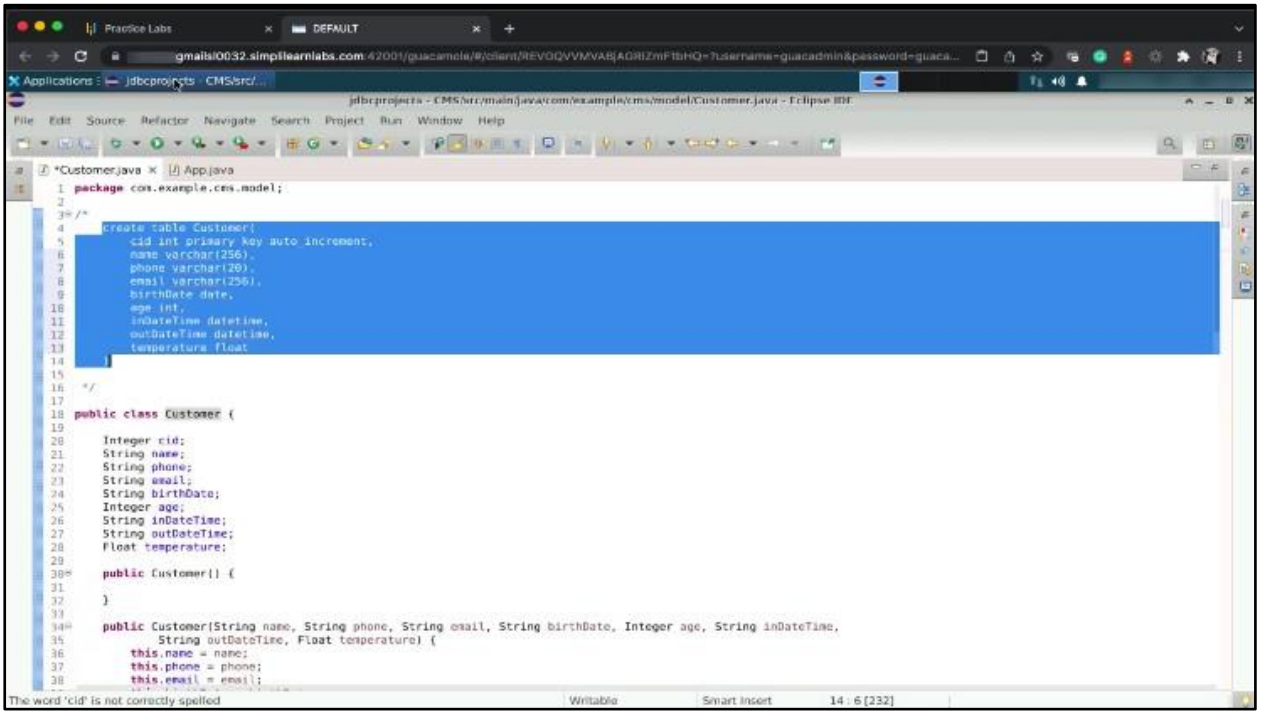
2.10 Set the **birthday** attribute with the datatype **date** as shown in the screenshot below:



2.11 Add a floating-point attribute for **temperature** and a datetime attribute for **outDataTime** as shown in the screenshot below:

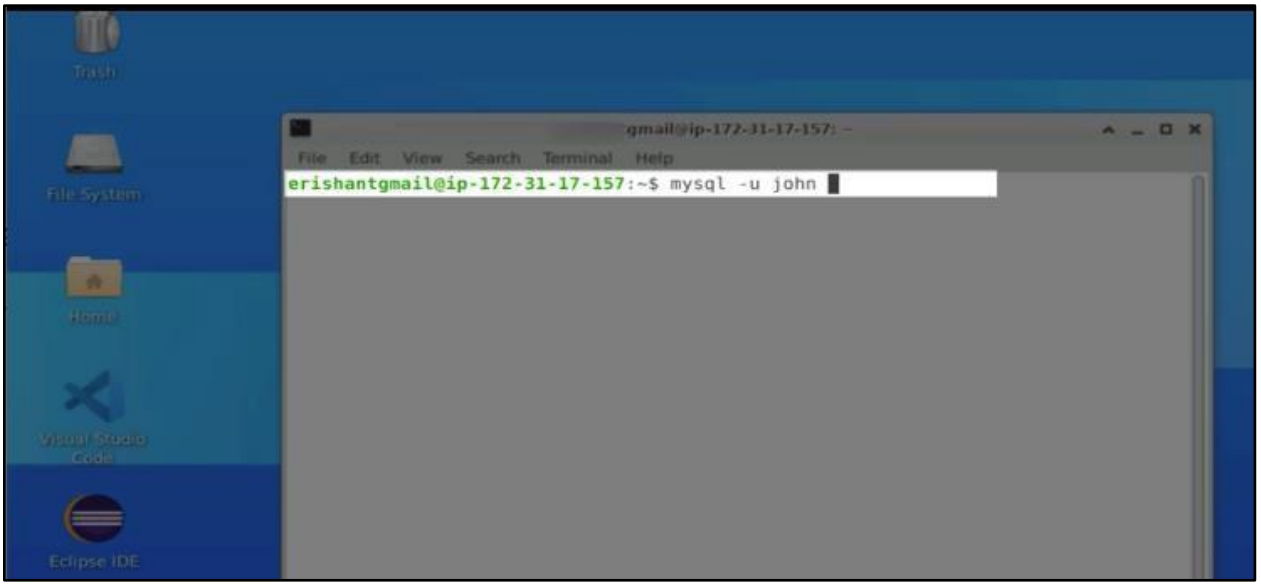


2.12 Copy the entire **Customer** table query

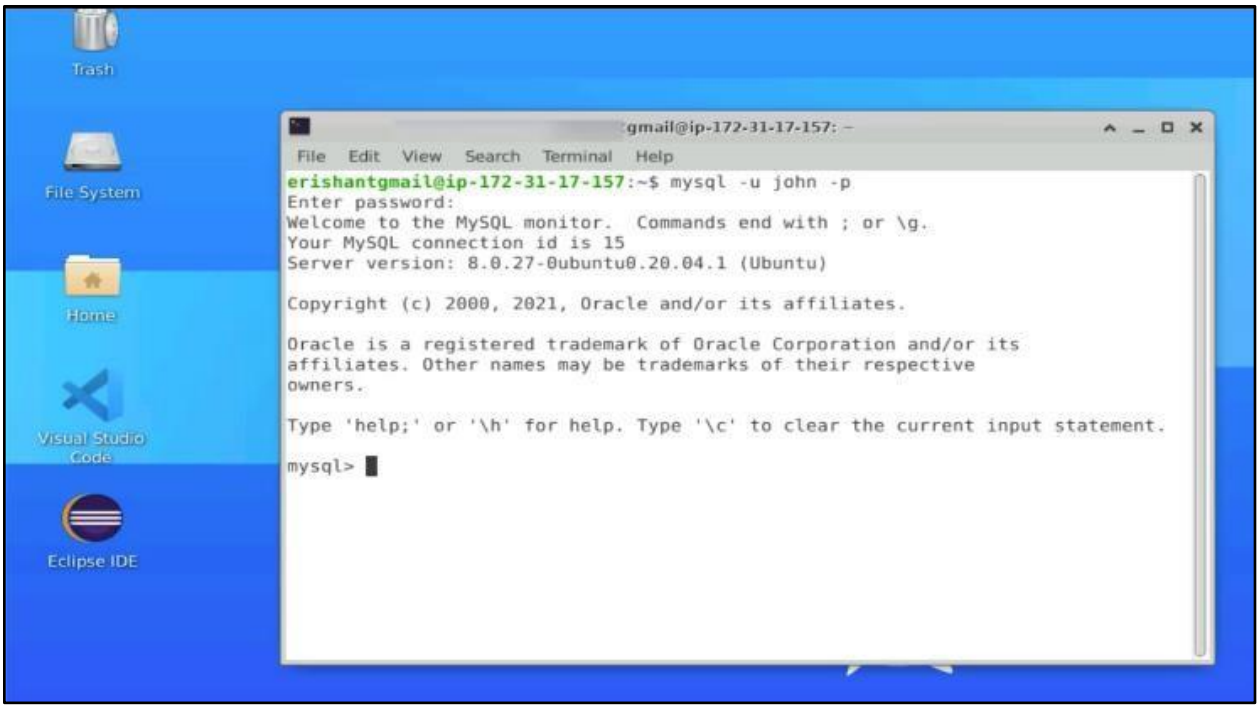


Step 3: Open the terminal window

3.1 Open the terminal window and log in to MySQL using the command **mysql -u john -p**

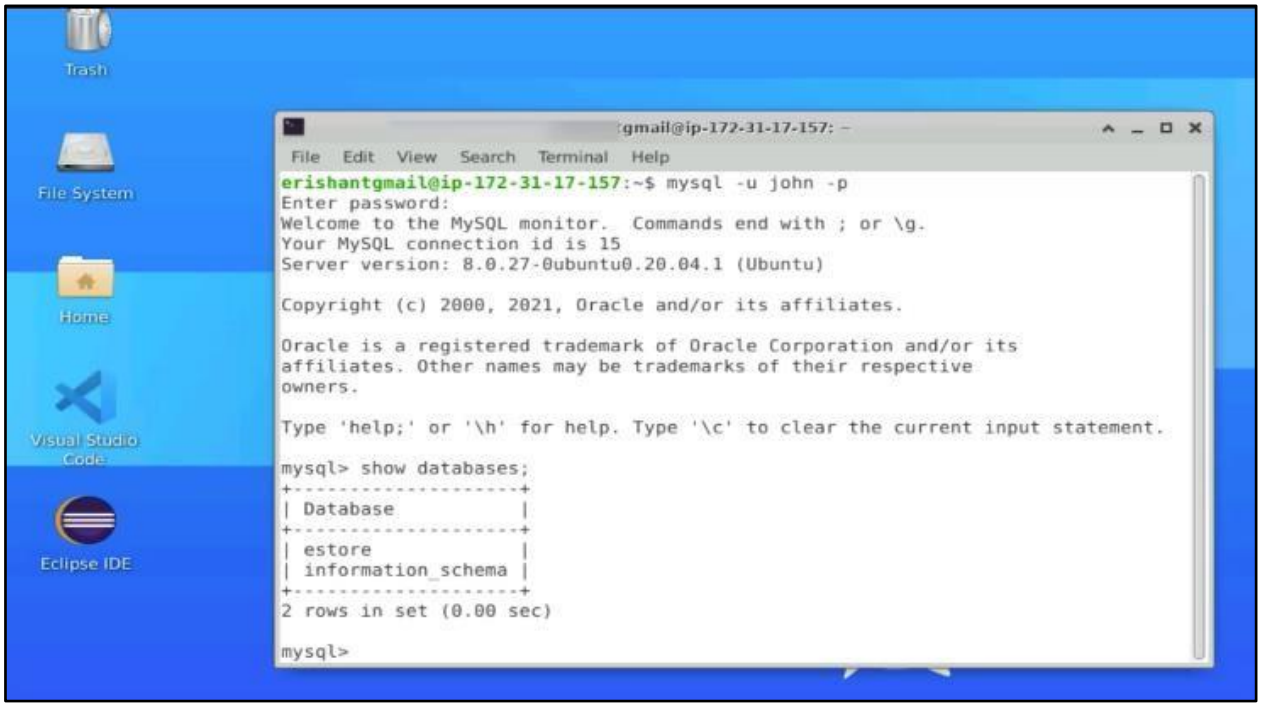


3.2 Enter the **password**

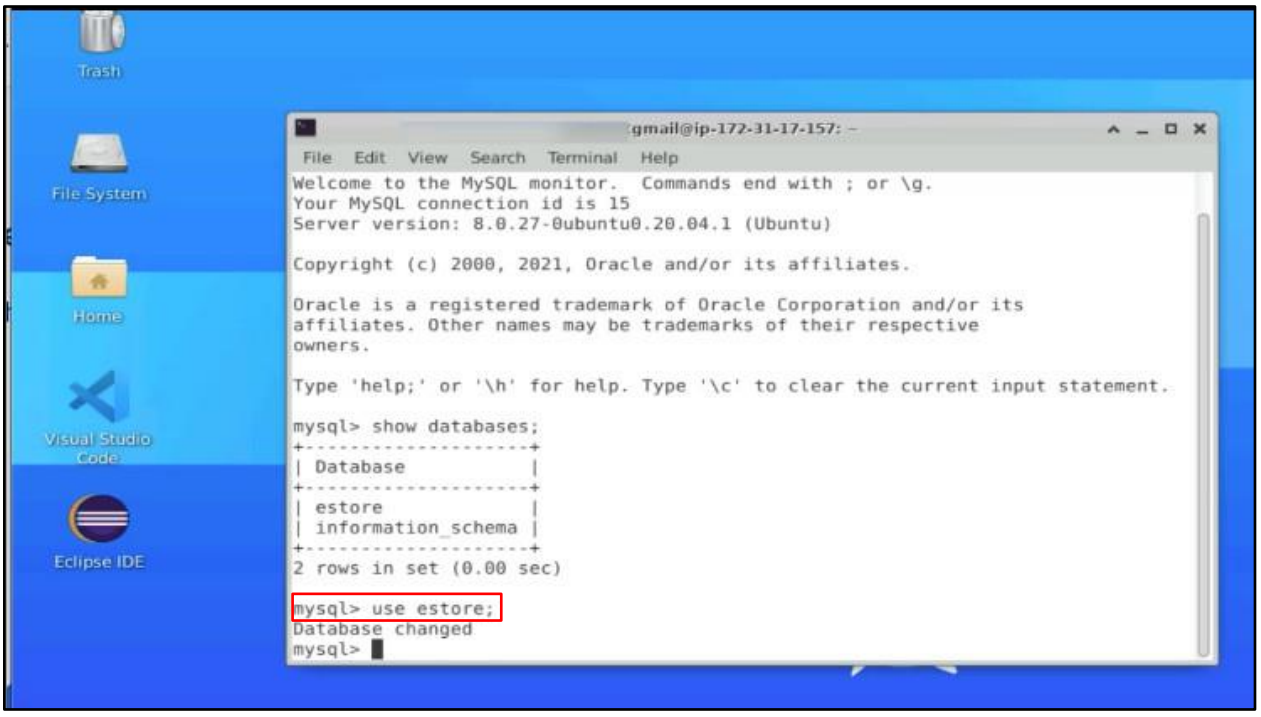


Note: A user named **John** has already been created for the database.

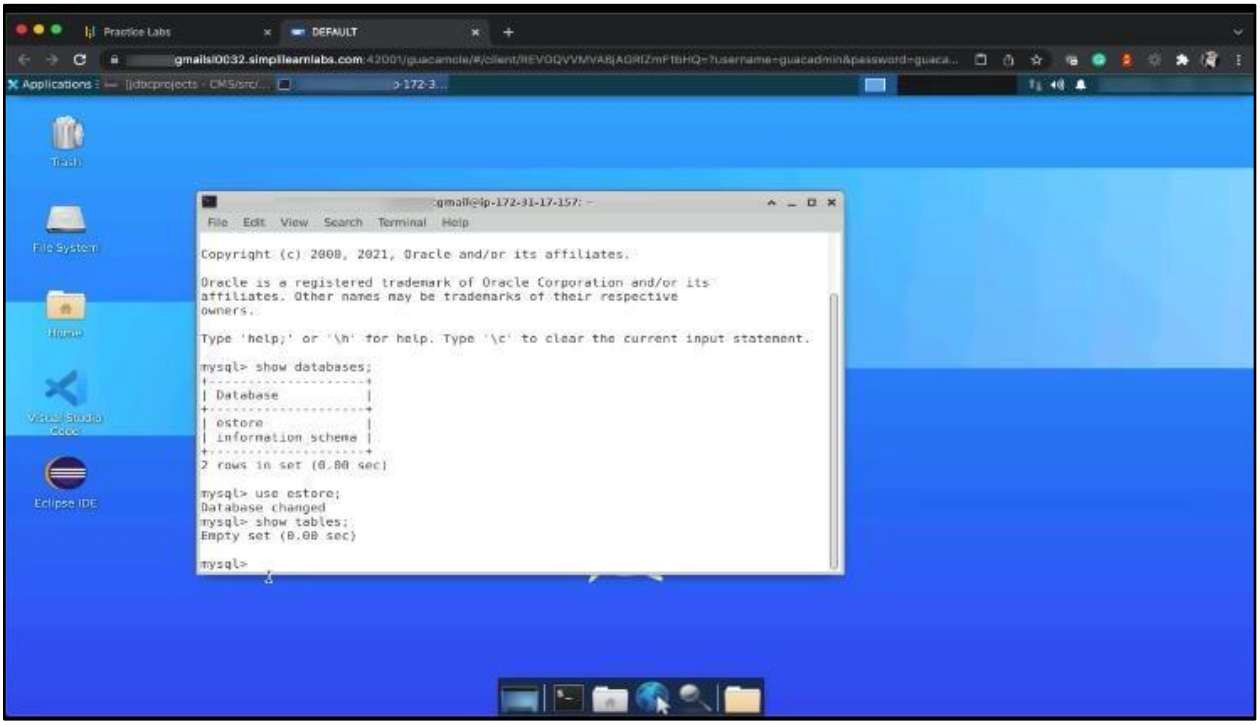
3.3 Type the **show databases;** command to check the available database names as shown in the screenshot below:



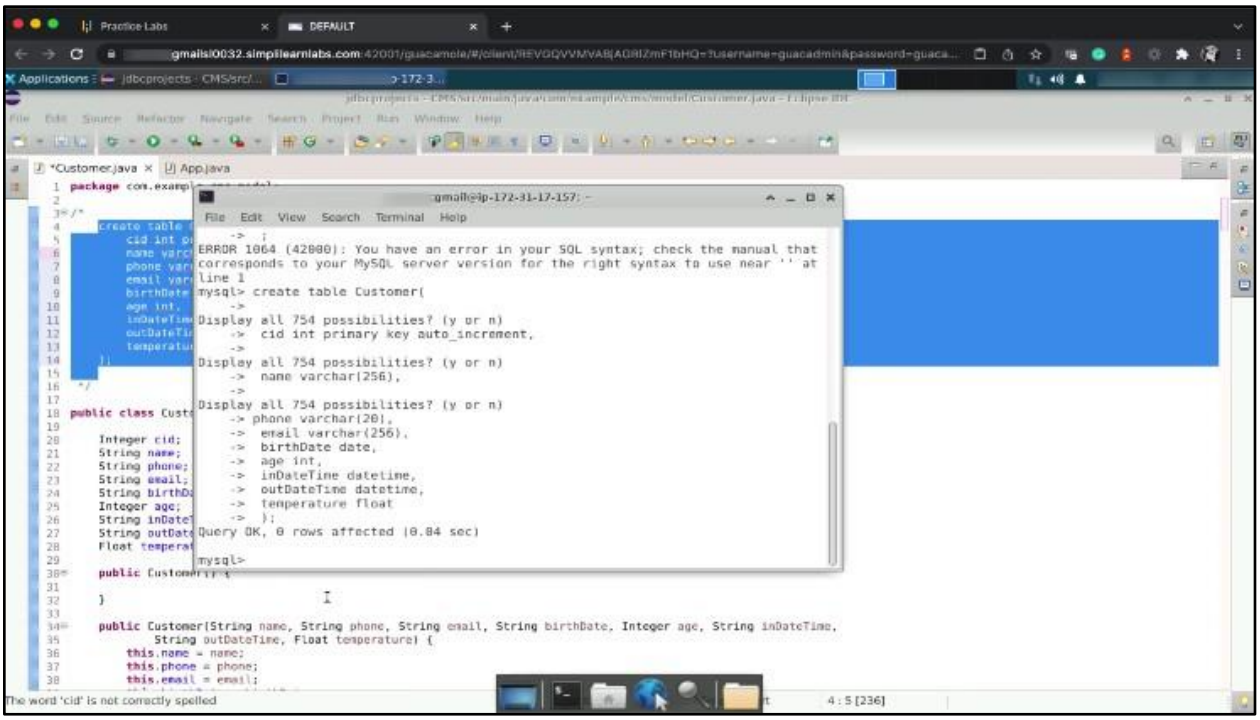
3.4 Enter **use estore;** to work on the estore databases shown in the screenshot below:



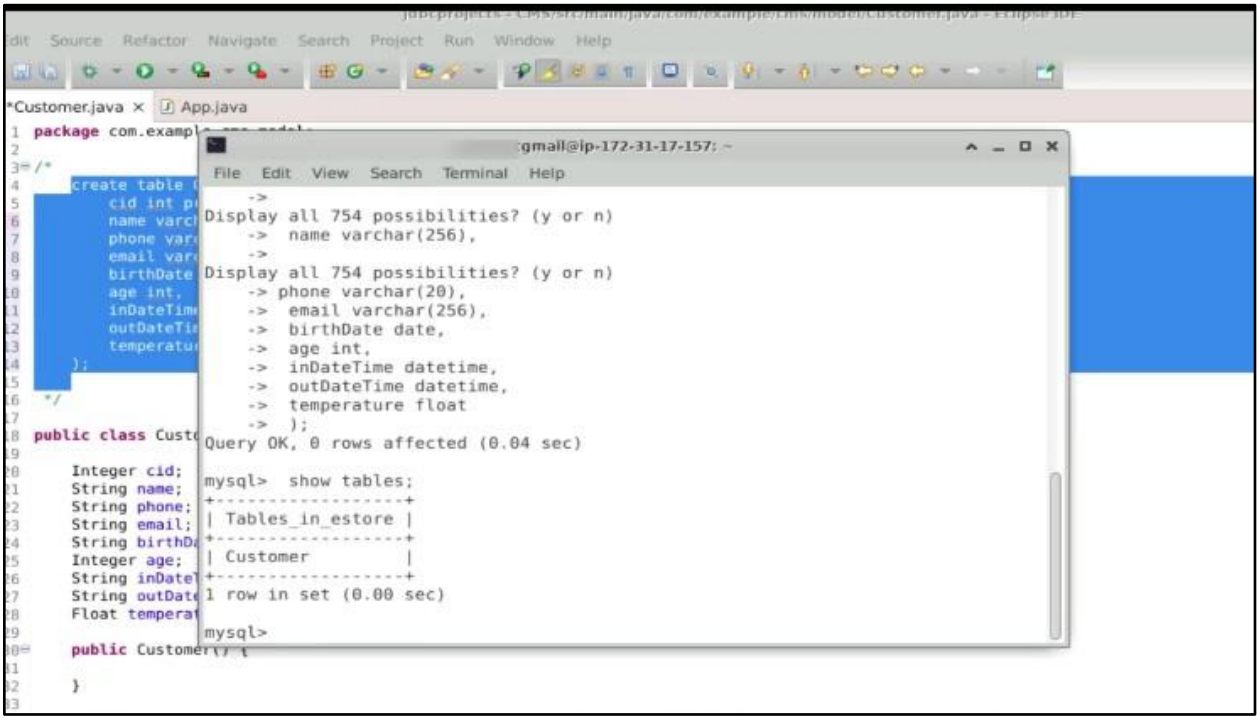
3.5 Enter **show tables;** to check tables available in the estore database



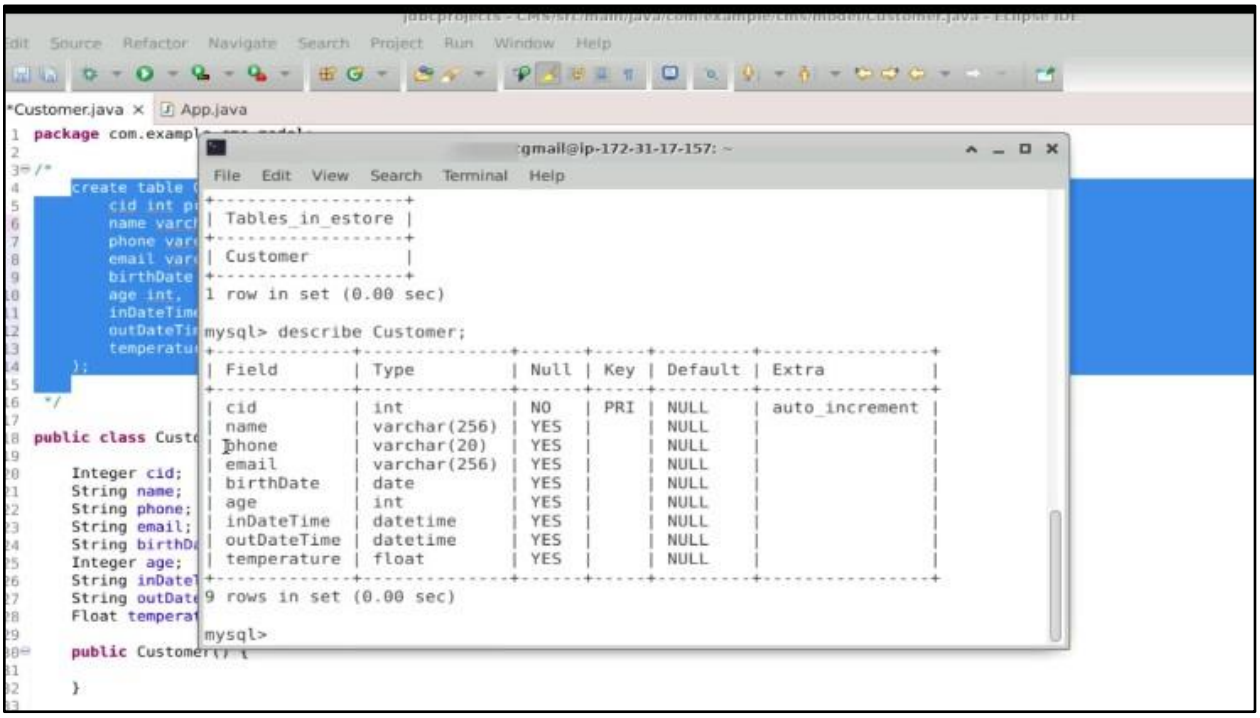
3.6 Paste the **Customer** table copied in step 2.12



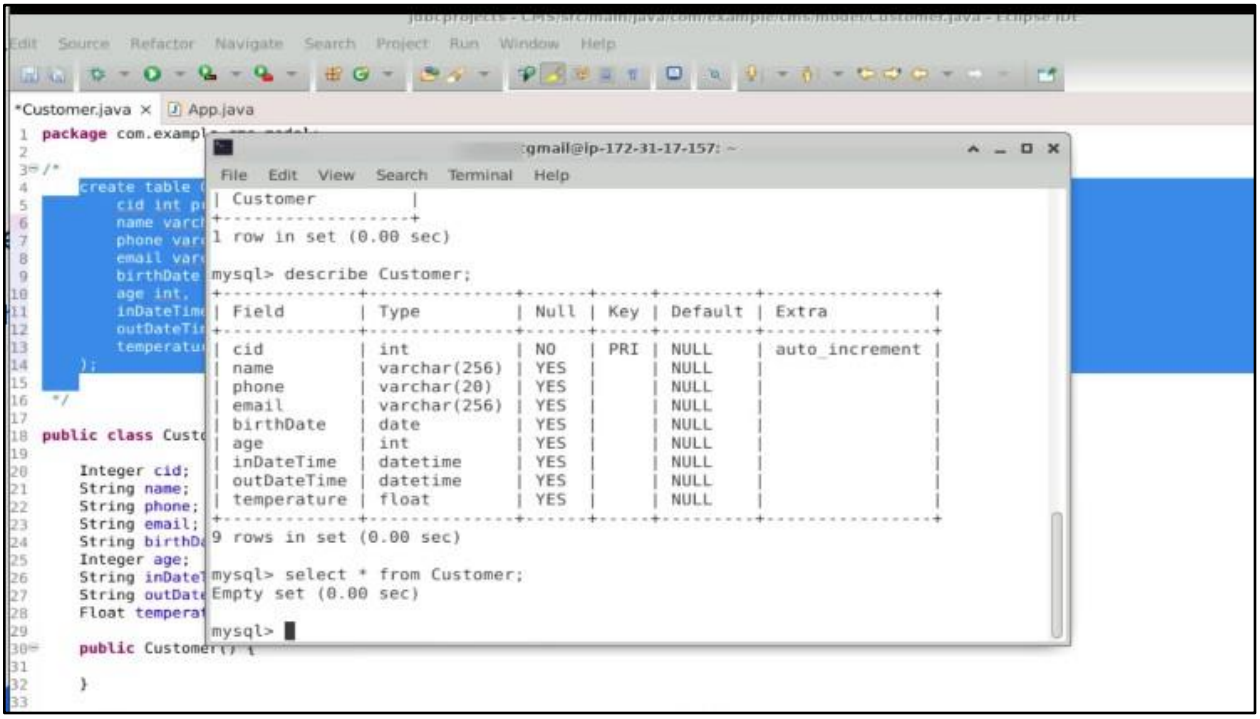
3.7 Run the **show tables;** command to see the changes



3.8 Enter the **describe Customer;** command to view the customer table details

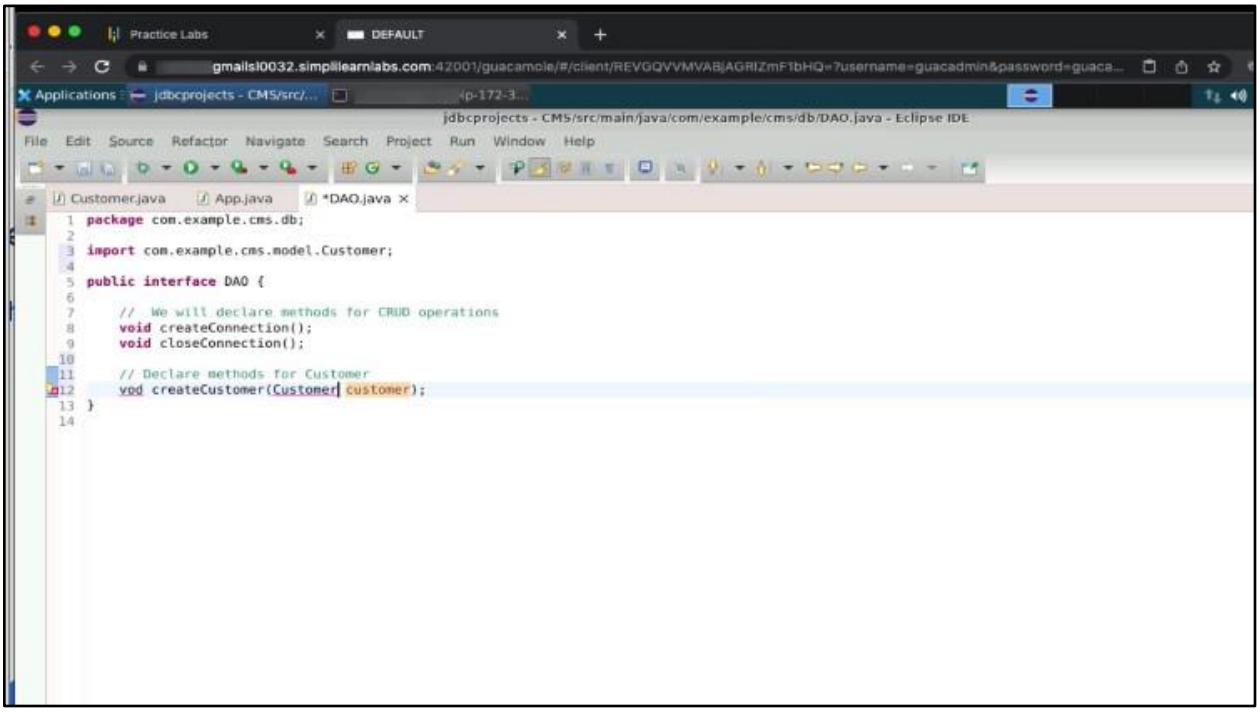


3.9 Enter **select * from Customer;** to check the data in the table

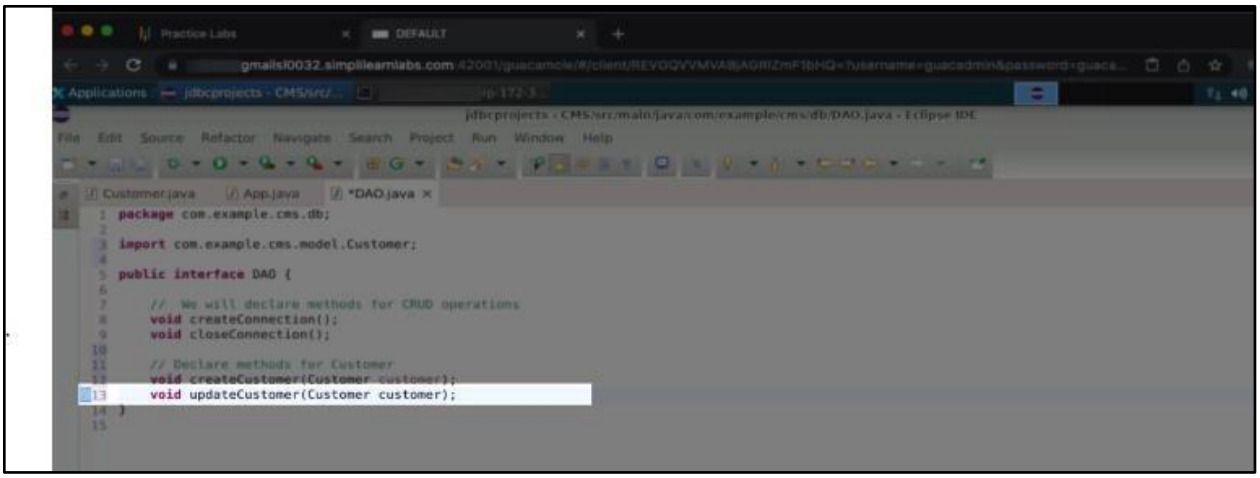


Step 4: Declare a method for a customer in the DAO interface

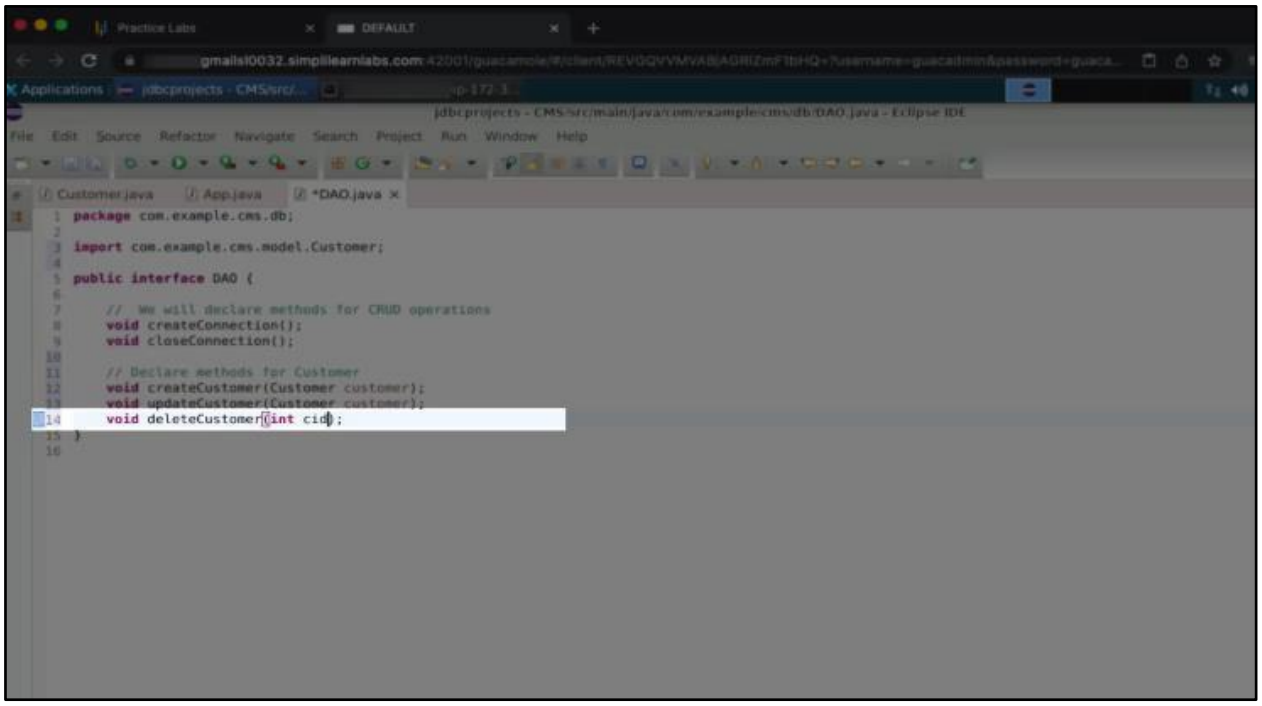
4.1 Declare methods for a customer in the DAO interface. The first one is **createCustomer()** that takes a single customer as input.



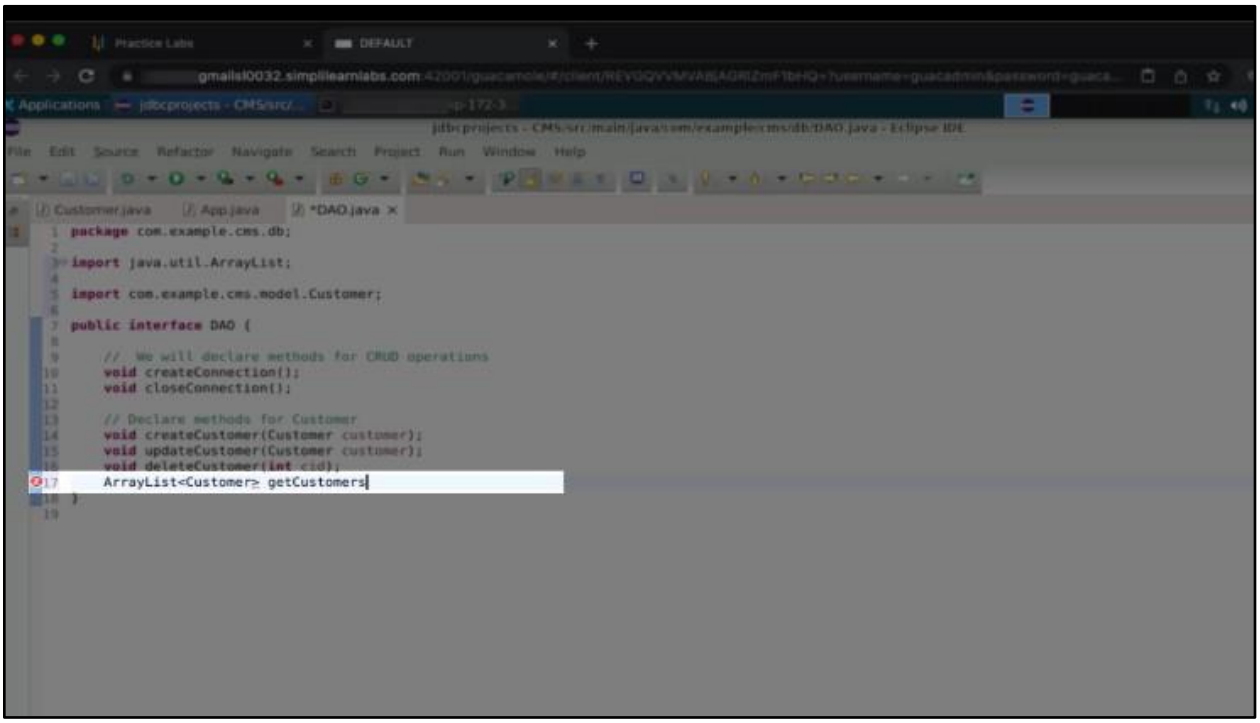
4.2 Additionally, implement **updateCustomer()** which also takes a customer as input for making updates



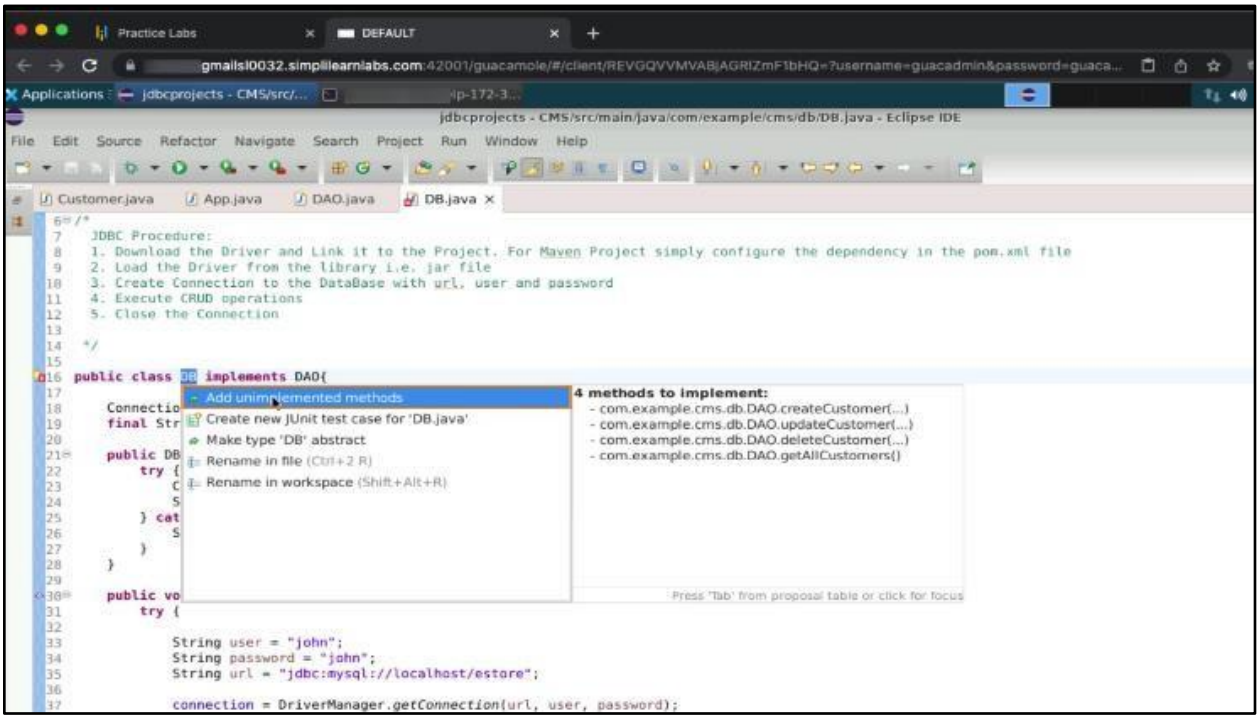
4.3 Create **deleteCustomer(),** which takes a single input **cid**



4.4 Implement the **getAllCustomers()** method to retrieve the list of customers



4.5 Navigate to the **DB.java** file where you can see the error, and click on the error to add the unimplemented methods



By following these steps, you have successfully created a model named **Customer** and established a corresponding table in the database using ORM. Additionally, we have implemented methods following the DAO design pattern to perform CRUD operations on the data.