

## Lesson 01 Demo 04

### Writing CRUD Operations with PreparedStatement API

**Objective:** To implement PreparedStatement API with JDBC for writing CRUD operations for managing database records efficiently and securely

**Tool required:** Eclipse IDE

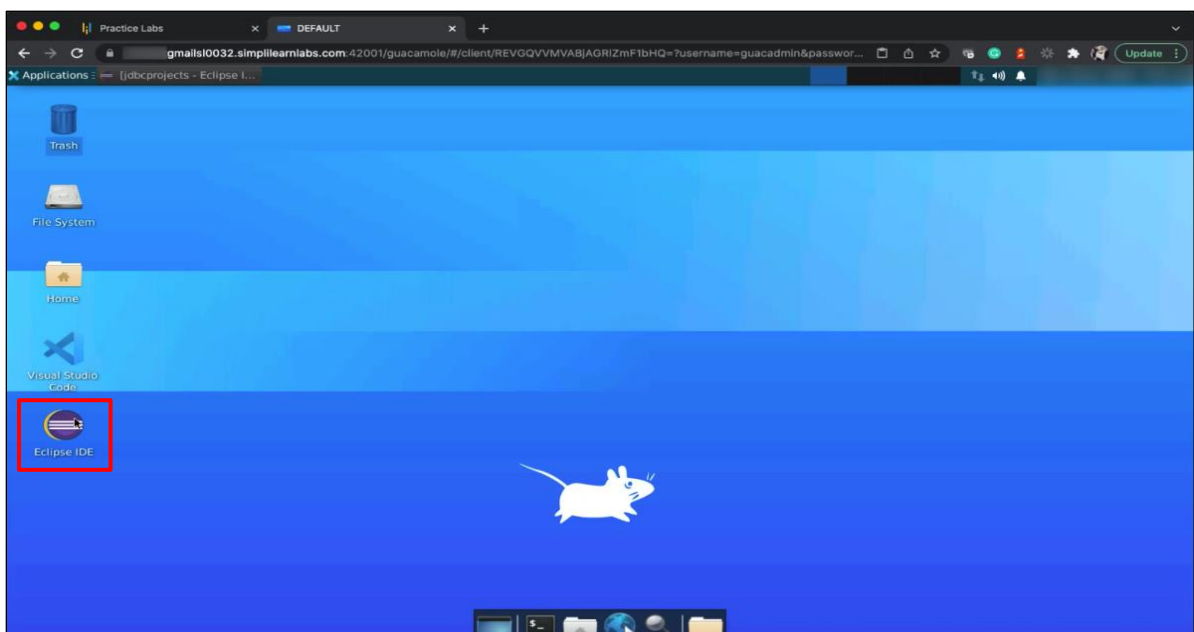
**Prerequisites:** Lesson 01 Demo 01

#### Steps to be followed:

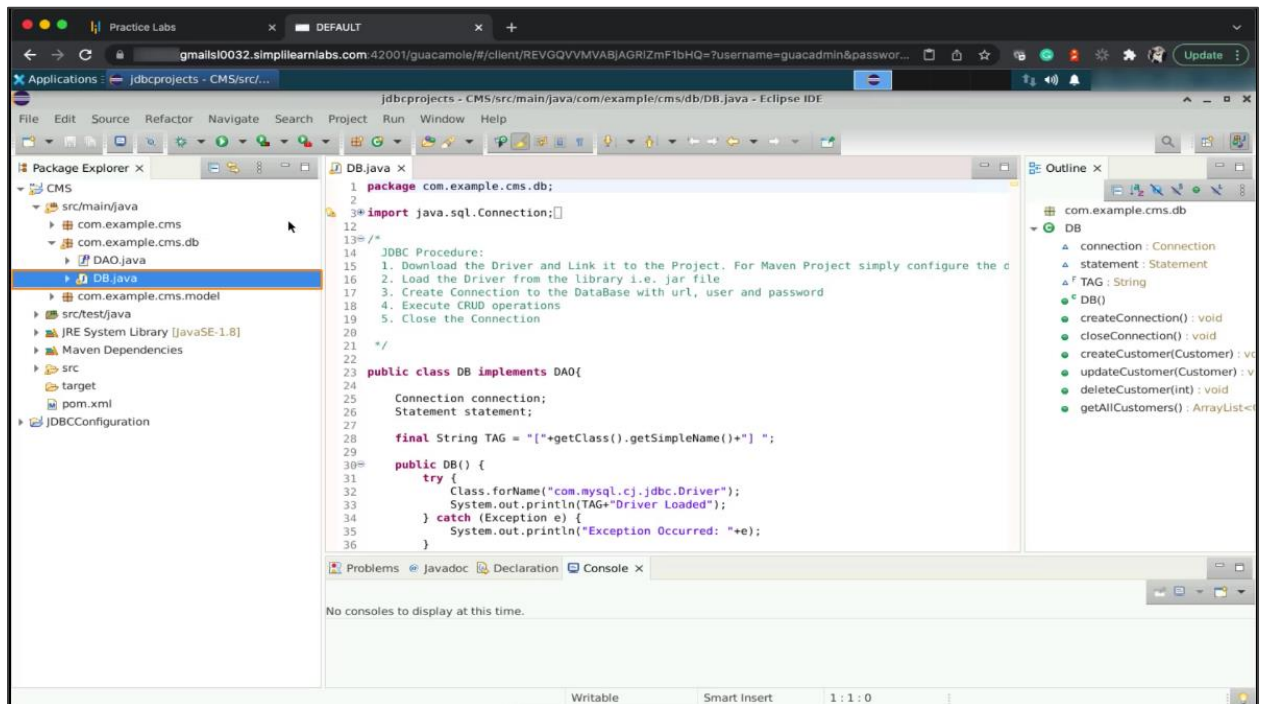
1. Perform the Create operation
2. Perform the Update operation
3. Perform the Delete operation
4. Perform the getAllCustomer operation

#### Step 1: Perform the Create operation

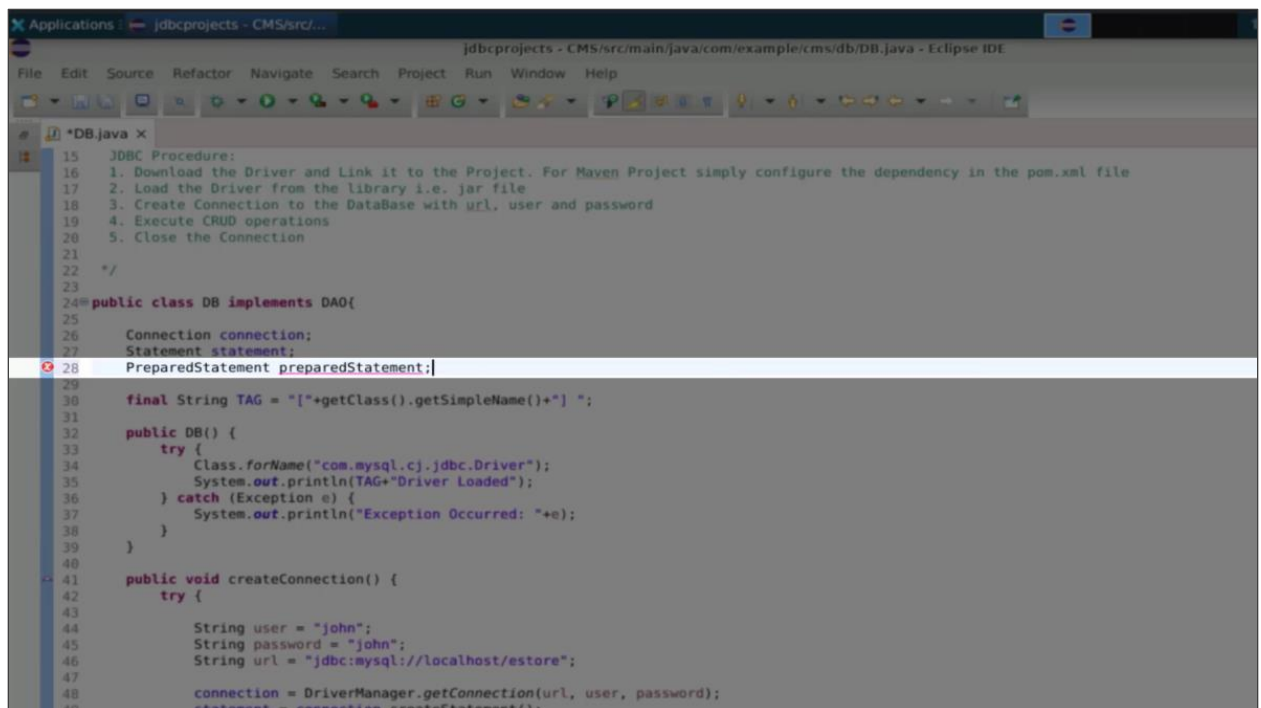
##### 1.1 Open Eclipse IDE



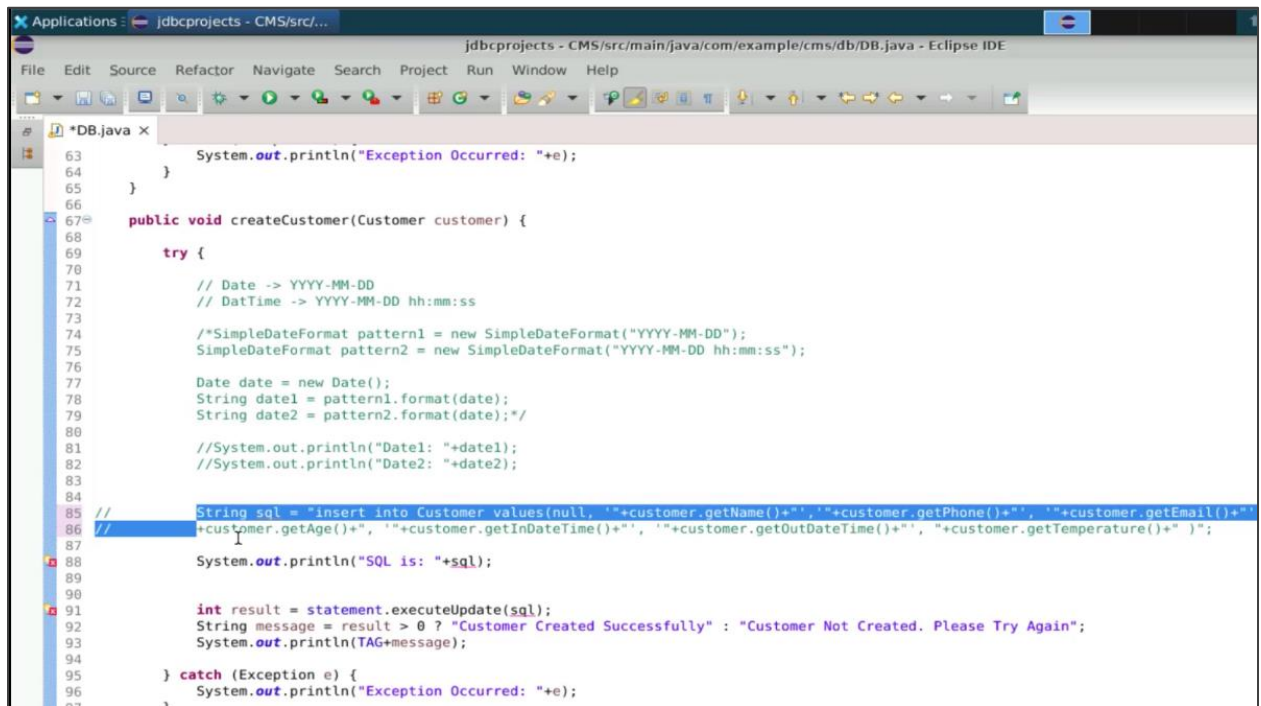
## 1.2 Open the DB.java file in the CMS project folder



## 1.3 Write another API for PreparedStatement with a reference variable preparedStatement;



## 1.4 Comment out the previous SQL statement

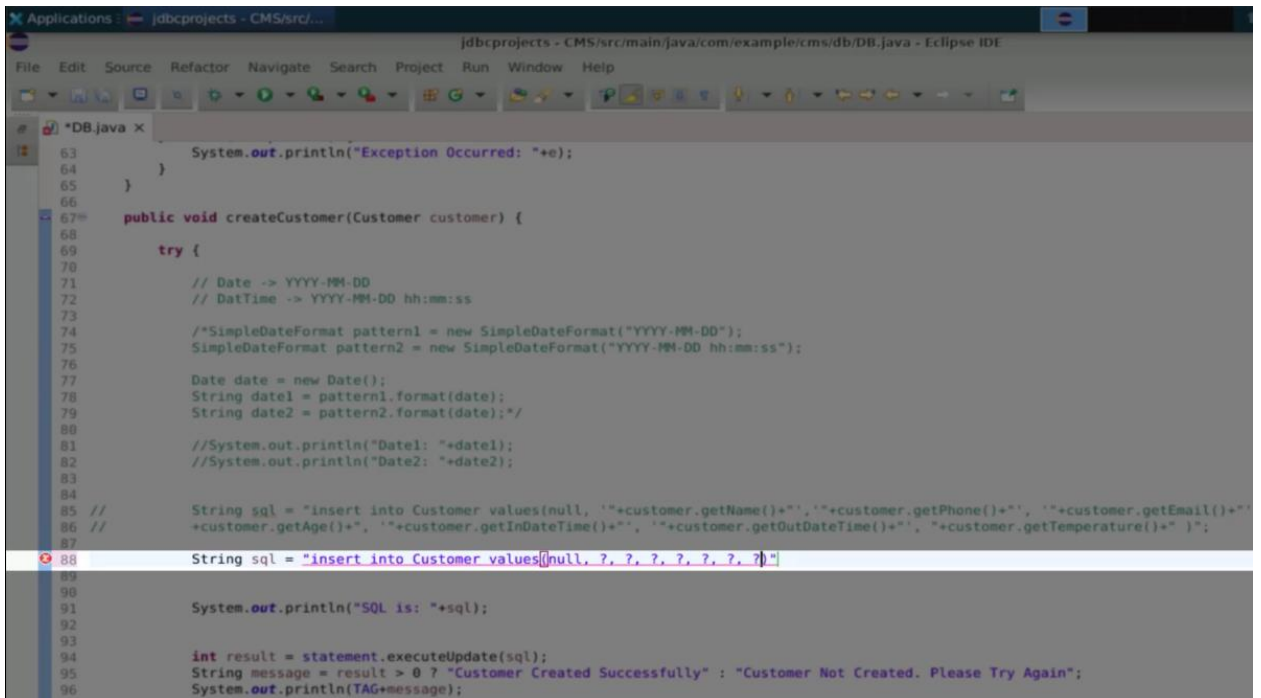


```

63         System.out.println("Exception Occurred: "+e);
64     }
65 }
66
67 public void createCustomer(Customer customer) {
68     try {
69         // Date -> YYYY-MM-DD
70         // DateTime -> YYYY-MM-DD hh:mm:ss
71
72         /*SimpleDateFormat pattern1 = new SimpleDateFormat("YYYY-MM-DD");
73         SimpleDateFormat pattern2 = new SimpleDateFormat("YYYY-MM-DD hh:mm:ss");
74
75         Date date = new Date();
76         String date1 = pattern1.format(date);
77         String date2 = pattern2.format(date);*/
78
79         //System.out.println("Date1: "+date1);
80         //System.out.println("Date2: "+date2);
81
82         //String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"'
83         //"+customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+"' )";
84
85         System.out.println("SQL is: "+sql);
86
87         int result = statement.executeUpdate(sql);
88         String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
89         System.out.println(TAG+message);
90     } catch (Exception e) {
91         System.out.println("Exception Occurred: "+e);
92     }
93 }

```

## 1.5 Write the SQL statement for the insert operation in the Customer table

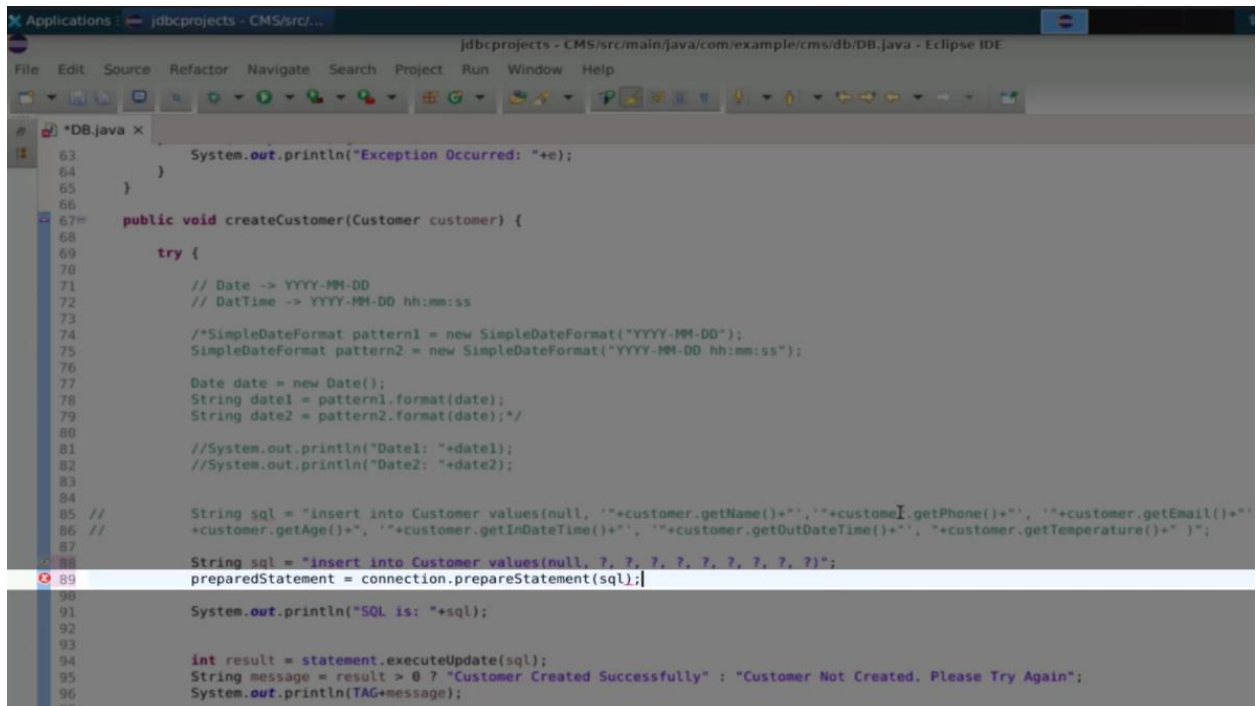


```

63         System.out.println("Exception Occurred: "+e);
64     }
65 }
66
67 public void createCustomer(Customer customer) {
68     try {
69         // Date -> YYYY-MM-DD
70         // DateTime -> YYYY-MM-DD hh:mm:ss
71
72         /*SimpleDateFormat pattern1 = new SimpleDateFormat("YYYY-MM-DD");
73         SimpleDateFormat pattern2 = new SimpleDateFormat("YYYY-MM-DD hh:mm:ss");
74
75         Date date = new Date();
76         String date1 = pattern1.format(date);
77         String date2 = pattern2.format(date);*/
78
79         //System.out.println("Date1: "+date1);
80         //System.out.println("Date2: "+date2);
81
82         //String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"'
83         //"+customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+"' )";
84
85         System.out.println("SQL is: "+sql);
86
87         int result = statement.executeUpdate(sql);
88         String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
89         System.out.println(TAG+message);
90     } catch (Exception e) {
91         System.out.println("Exception Occurred: "+e);
92     }
93 }

```

## 1.6 Initialize the preparedStatement

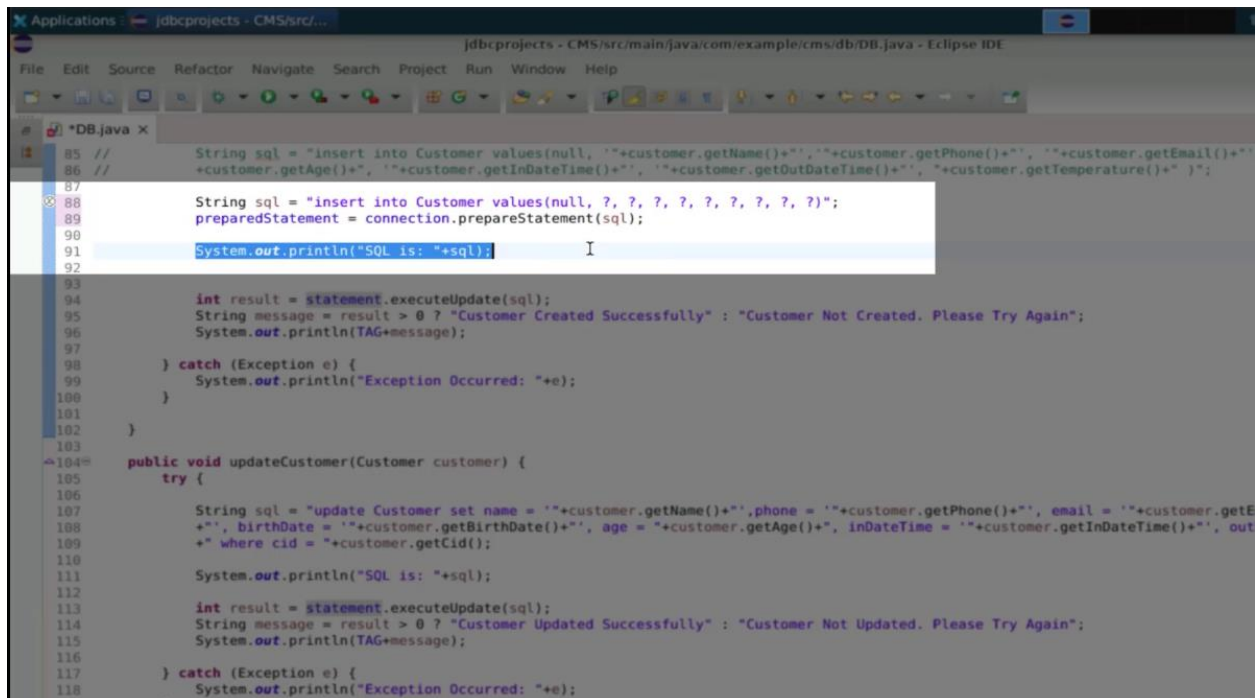


```

63         System.out.println("Exception Occurred: "+e);
64     }
65 }
66
67 public void createCustomer(Customer customer) {
68     try {
69         // Date -> YYYY-MM-DD
70         // DateTime -> YYYY-MM-DD hh:mm:ss
71
72         // SimpleDateFormat pattern1 = new SimpleDateFormat("YYYY-MM-DD");
73         SimpleDateFormat pattern2 = new SimpleDateFormat("YYYY-MM-DD hh:mm:ss");
74
75         Date date = new Date();
76         String date1 = pattern1.format(date);
77         String date2 = pattern2.format(date);
78
79         //System.out.println("Date1: "+date1);
80         //System.out.println("Date2: "+date2);
81
82         String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"',
83         // "+customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+" )";
84
85         String sql = "insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?, ?)";
86         preparedStatement = connection.prepareStatement(sql);
87
88         System.out.println("SQL is: "+sql);
89
90         int result = statement.executeUpdate(sql);
91         String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
92         System.out.println(TAG+message);
93     }
94 }

```

## 1.7 Cut (Ctrl+X) the selected line of code



```

85 // String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"',
86 // "+customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+" )";
87
88 String sql = "insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?, ?)";
89 preparedStatement = connection.prepareStatement(sql);
90
91 System.out.println("SQL is: "+sql);
92
93
94 int result = statement.executeUpdate(sql);
95 String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
96 System.out.println(TAG+message);
97
98 } catch (Exception e) {
99     System.out.println("Exception Occurred: "+e);
100 }
101
102 }
103
104 public void updateCustomer(Customer customer) {
105     try {
106
107         String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getE
108         +"'', birthDate = '"+customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', out
109         +"' where cid = '"+customer.getCid()+'";
110
111         System.out.println("SQL is: "+sql);
112
113         int result = statement.executeUpdate(sql);
114         String message = result > 0 ? "Customer Updated Successfully" : "Customer Not Updated. Please Try Again";
115         System.out.println(TAG+message);
116
117     } catch (Exception e) {
118         System.out.println("Exception Occurred: "+e);
119     }
120 }

```

1.8 Paste the code above the preparedStatement object (line 89), and now pass the String sql

```

85 //      String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"'
86 //      +customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+"' )";
87
88      String sql = "insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?, ?)";
89      System.out.println("SQL is: "+sql);
90      preparedStatement = connection.prepareStatement(sql);
91
92
93
94
95      int result = statement.executeUpdate(sql);
96      String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
97      System.out.println(TAG+message);
98
99  } catch (Exception e) {
100      System.out.println("Exception Occurred: "+e);
101  }
102
103  }
104
105  public void updateCustomer(Customer customer) {
106      try {
107
108          String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getE
109          +", birthDate = '"+customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', out
110          + " where cid = '"+customer.getCid()";
111
112          System.out.println("SQL is: "+sql);
113
114          int result = statement.executeUpdate(sql);
115          String message = result > 0 ? "Customer Updated Successfully" : "Customer Not Updated. Please Try Again";
116          System.out.println(TAG+message);
117
118      } catch (Exception e) {

```

1.9 Substitute the data through the **getter** method, use the **setString()** method and index starting from 1 for each attribute of data, as shown below from line 92 to line 99:

```

85 //      String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"'
86 //      +customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+"' )";
87
88      String sql = "insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?, ?)";
89      System.out.println("SQL is: "+sql);
90      preparedStatement = connection.prepareStatement(sql);
91
92      preparedStatement.setString(1, customer.getName());
93      preparedStatement.setString(2, customer.getPhone());
94      preparedStatement.setString(3, customer.getEmail());
95      preparedStatement.setString(4, customer.getBirthDate());
96      preparedStatement.setInt(5, customer.getAge());
97      preparedStatement.setString(6, customer.getInDateTime());
98      preparedStatement.setString(7, customer.getOutDateTime());
99      preparedStatement.setFloat(8, customer.getTemperature());
100
101
102      int result = statement.executeUpdate(sql);
103      String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
104      System.out.println(TAG+message);
105
106  } catch (Exception e) {
107      System.out.println("Exception Occurred: "+e);
108  }
109
110  }

```



1.10 Comment on the calling of the **executeUpdate()** function, as it will remain similar to the above lines

```

85 // String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"', '"+customer.getBirthDate()
86 // +customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+"' ";
87
88 String sql = "insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?)";
89 System.out.println("SQL is: "+sql);
90
91 PreparedStatement = connection.prepareStatement(sql);
92 PreparedStatement.setString(1, customer.getName());
93 PreparedStatement.setString(2, customer.getPhone());
94 PreparedStatement.setString(3, customer.getEmail());
95 PreparedStatement.setString(4, customer.getBirthDate());
96 PreparedStatement.setInt(5, customer.getAge());
97 PreparedStatement.setString(6, customer.getInDateTime());
98 PreparedStatement.setString(7, customer.getOutDateTime());
99 PreparedStatement.setFloat(8, customer.getTemperature());
100
101 //int result = statement.executeUpdate(sql);
102 String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
103 System.out.println(TAG+message);
104
105 } catch (Exception e) {
106     System.out.println("Exception Occurred: "+e);
107 }
108
109 }
110
111
112 public void updateCustomer(Customer customer) {
113     try {
114
115         String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getEmail()
116         +customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', outDateTime = '"+customer.getO
117         +" where cid = '"+customer.getCid();
118

```

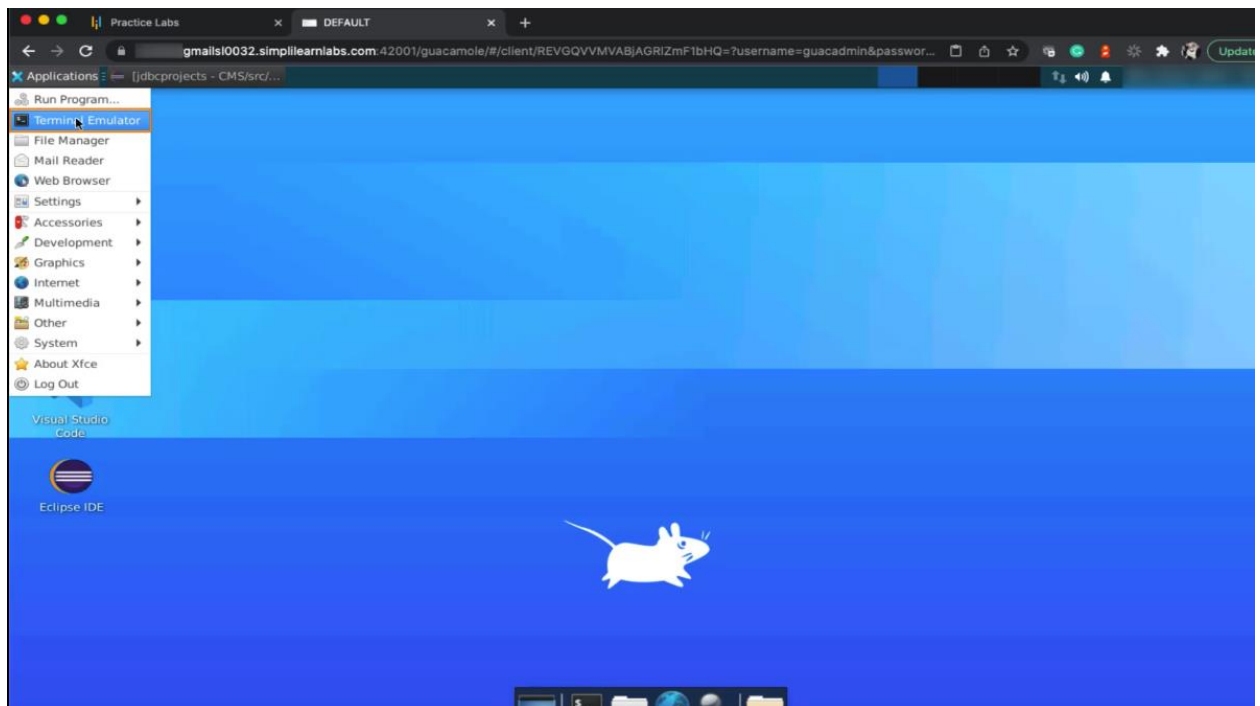
1.11 To execute the method, use **executeUpdate()**

```

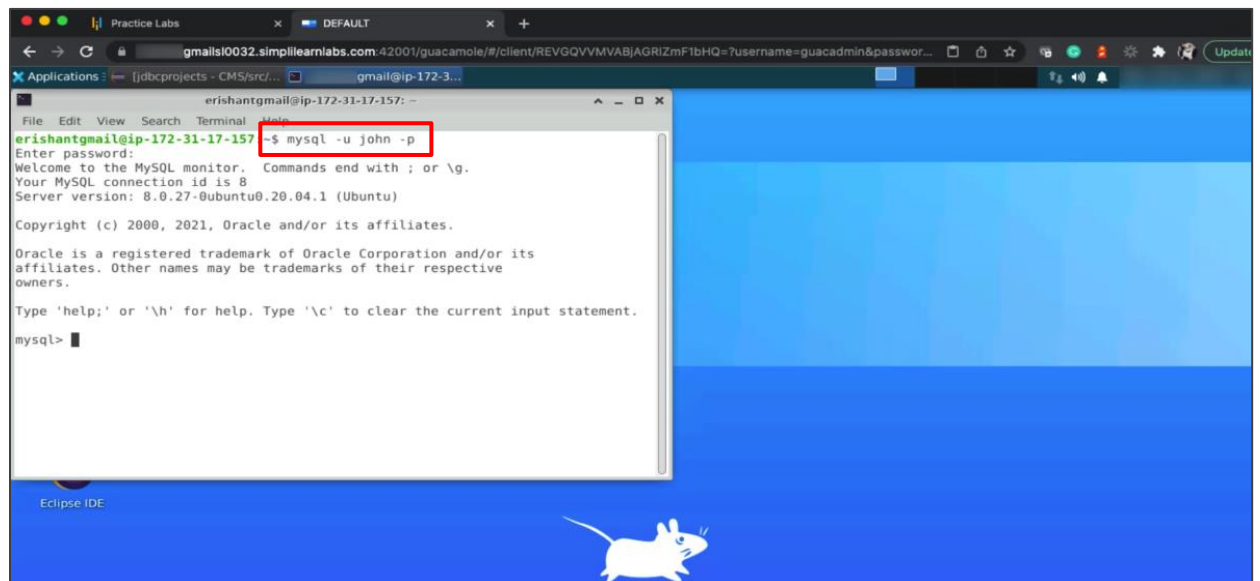
85 // String sql = "insert into Customer values(null, '"+customer.getName()+"', '"+customer.getPhone()+"', '"+customer.getEmail()+"', '"+customer.getBirthDate()
86 // +customer.getAge()+"', '"+customer.getInDateTime()+"', '"+customer.getOutDateTime()+"', '"+customer.getTemperature()+"' ";
87
88 String sql = "insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?)";
89 System.out.println("SQL is: "+sql);
90
91 PreparedStatement = connection.prepareStatement(sql);
92 PreparedStatement.setString(1, customer.getName());
93 PreparedStatement.setString(2, customer.getPhone());
94 PreparedStatement.setString(3, customer.getEmail());
95 PreparedStatement.setString(4, customer.getBirthDate());
96 PreparedStatement.setInt(5, customer.getAge());
97 PreparedStatement.setString(6, customer.getInDateTime());
98 PreparedStatement.setString(7, customer.getOutDateTime());
99 PreparedStatement.setFloat(8, customer.getTemperature());
100
101 //int result = statement.executeUpdate(sql);
102 String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
103 System.out.println(TAG+message);
104
105 } catch (Exception e) {
106     System.out.println("Exception Occurred: "+e);
107 }
108
109 }
110
111
112 public void updateCustomer(Customer customer) {
113     try {
114
115         String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getEmail()
116         +customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', outDateTime = '"+customer.getO
117         +" where cid = '"+customer.getCid();
118
119         System.out.println("SQL is: "+sql);
120
121         int result = statement.executeUpdate(sql);
122

```

## 1.12 Open the terminal

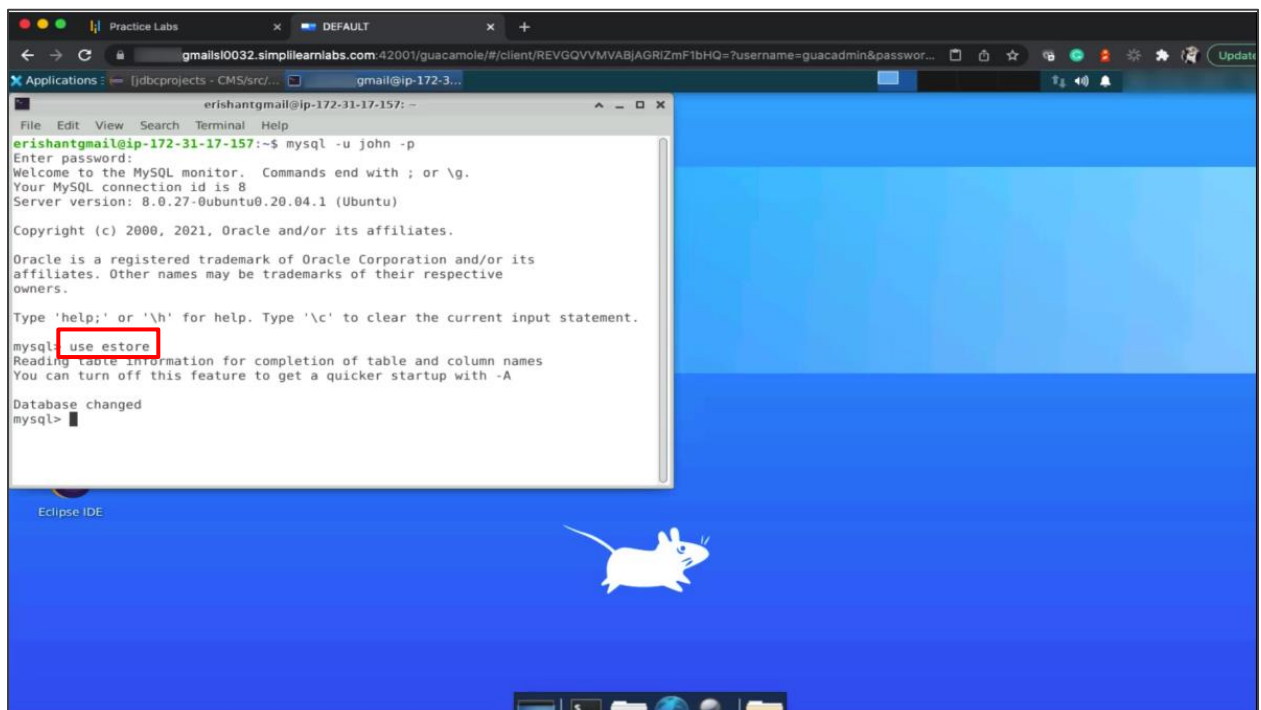


## 1.13 Log in into SQL using the command `mysql -u john -p` and the password is `john`.



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

## 1.14 Run command **use estore;** to change the database



```

erishantgmail@ip-172-31-17-157: ~
erishantgmail@ip-172-31-17-157:~$ mysql -u john -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

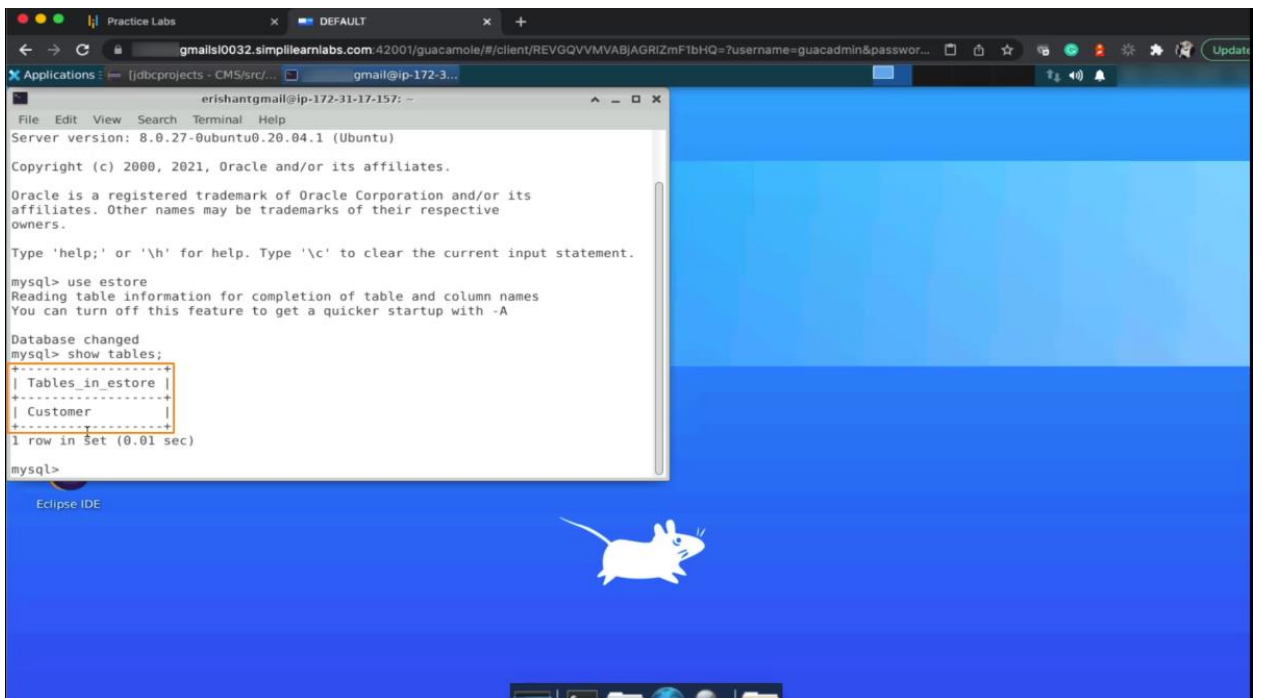
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use estore
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
  
```

## 1.15 Run the **show tables;** command to check available tables in the **estore** database



```

erishantgmail@ip-172-31-17-157: ~
erishantgmail@ip-172-31-17-157:~$ mysql -u john -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 8.0.27-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use estore
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_estore |
+-----+
| Customer          |
+-----+
1 row in set (0.01 sec)

mysql>
  
```



## 1.16 Run the **describe Customer;** command to see table details

```

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 (0.01 sec)

mysql>
  
```

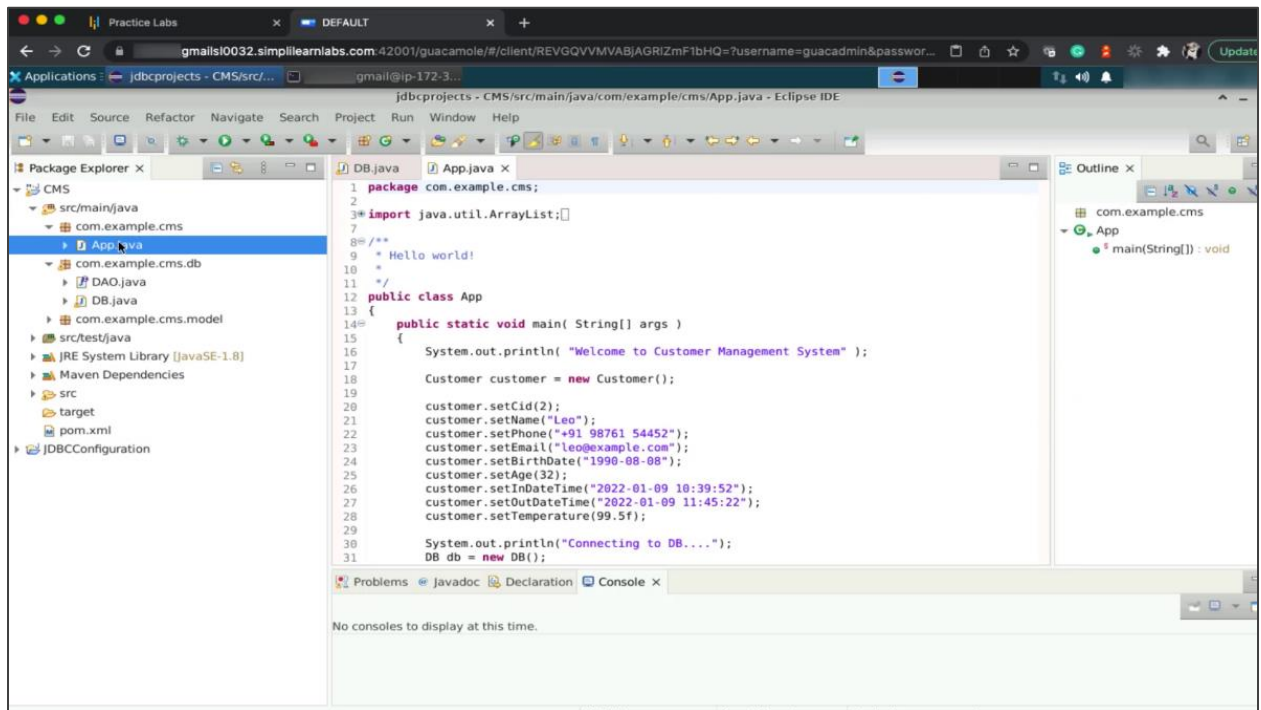
## 1.17 Run the select command as **select \* from Customer;** to see data from the table

```

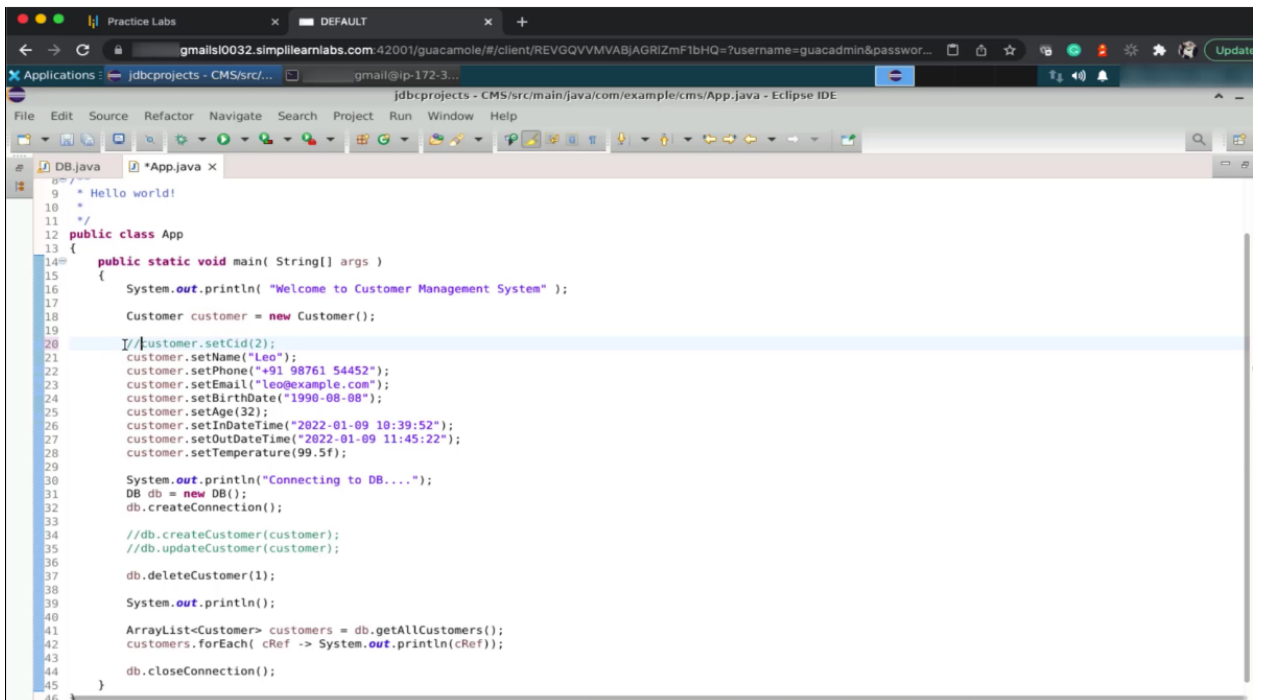
mysql> select * from Customer;
+-----+-----+-----+-----+-----+-----+-----+-----+
| cid | name | phone | email | birthDate | age | inDateTime | outDateTime | temperature |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 2 | John Watson | +91 98761 22222 | john.watson@example.com | 1990-08-08 | 32 | 2022-01-08 10:39:52 | 2022-01-08 11:45:22 | 98.5 |
| 3 | Leo | +91 98761 54452 | leo@example.com | 1990-08-08 | 32 | 2022-01-09 10:39:52 | 2022-01-09 11:45:22 | 99.5 |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
  
```

## 1.18 Go back to Eclipse and open the **App.java** file to use the insert query



## 1.19 Comment out **Cid** as it is an auto-increment



## 1.20 Change the details in the customer fields to create a new customer dataset

```

16      System.out.println("Welcome to Customer Management System" );
17
18      Customer customer = new Customer();
19
20      //customer.setCid(2);
21      customer.setName("George");
22      customer.setPhone("+91 98765 54452");
23      customer.setEmail("leo@example.com");
24      customer.setBirthDate("1990-08-08");
25      customer.setAge(32);
26      customer.setInDateTime("2022-01-09 10:39:52");
27      customer.setOutDateTime("2022-01-09 11:45:22");
28      customer.setTemperature(99.5f);
29
30      System.out.println("Connecting to DB....");
31      DB db = new DB();
32      db.createConnection();
33

```

```

19
20      //customer.setCid(2);
21      customer.setName("George");
22      customer.setPhone("+91 99009 54452");
23      customer.setEmail("leo@example.com");
24      customer.setBirthDate("1990-08-08");
25      customer.setAge(32);
26      customer.setInDateTime("2022-01-09 10:39:52");
27      customer.setOutDateTime("2022-01-09 11:45:22");
28      customer.setTemperature(99.5f);
29
30      System.out.println("Connecting to DB....");
31      DB db = new DB();
32      db.createConnection();
33

```

```

19
20      //customer.setCid(2);
21      customer.setName("George");
22      customer.setPhone("+91 99009 88088");
23      customer.setEmail("george@example.com");
24      customer.setBirthDate("1990-08-08");
25      customer.setAge(32);
26      customer.setInDateTime("2022-01-09 10:39:52");
27      customer.setOutDateTime("2022-01-09 11:45:22");
28      customer.setTemperature(99.5f);
29
30      System.out.println("Connecting to DB....");
31      DB db = new DB();
32      db.createConnection();
33

```

```

19
20      //customer.setCid(2);
21      customer.setName("George");
22      customer.setPhone("+91 99009 88088");
23      customer.setEmail("george@example.com");
24      customer.setBirthDate("1986-04-04");
25      customer.setAge(32);
26      customer.setInDateTime("2022-01-09 10:39:52");
27      customer.setOutDateTime("2022-01-09 11:45:22");
28      customer.setTemperature(99.5f);
29
30      System.out.println("Connecting to DB....");
31      DB db = new DB();
32      db.createConnection();
33

```

```

19
20      //customer.setCid(2);
21      customer.setName("George");
22      customer.setPhone("+91 99009 88088");
23      customer.setEmail("george@example.com");
24      customer.setBirthDate("1986-04-04");
25      customer.setAge(32);
26      customer.setInDateTime("2022-01-12 10:39:52");
27      customer.setOutDateTime("2022-01-12 11:45:22");
28      customer.setTemperature(99.5f);
29
30      System.out.println("Connecting to DB....");
31      DB db = new DB();
32      db.createConnection();
33

```

## 1.21 Run the code, and the output can be seen in the console as **Customer Created Successfully**

The screenshot shows the Eclipse IDE with the following components:

- App.java:**

```

9  * Hello world!
10 *
11 */
12 public class App
13 {
14     public static void main( String[] args )
15     {
16         System.out.println( "Welcome to Customer Management System" );
17
18         Customer customer = new Customer();
19
20         //customer.setCid(2);
21         customer.setName("George");
22         customer.setPhone("+91 99009 88088");
23         customer.setEmail("george@example.com");
24         customer.setBirthDate("1986-04-04");
25         customer.setAge(32);
26         customer.setInDateTime("2022-01-12 10:39:52");
27         customer.setOutDateTime("2022-01-12 11:45:22");
28         customer.setTemperature(99.5f);
29
30         System.out.println("Connecting to DB...");
31         DB db = new DB();
32         db.createConnection();
33
34         db.createCustomer(customer);
35         //db.updateCustomer(customer);
36
37         //db.deleteCustomer(1);
38
39         //System.out.println();
40
41         //ArrayList<Customer> customers = db.getAllCustomers();
42         //customers.forEach( cRef -> System.out.println(cRef));
43
44         db.closeConnection();
45     }
46 }

```
- Console:**

```

<terminated> App [Java Application] [usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
Welcome to Customer Management System
Connecting to DB...
[DB] Driver Loaded
[DB] Connection Created
SQL is: insert into Customer values(null, ?, ?, ?, ?, ?, ?, ?)
[DB] Customer Created Successfully
[DB] Connection Closed. Close Status: true

```

## 1.22 Go back to the terminal and run the **select** command

The screenshot shows the Eclipse IDE with a terminal window open. The terminal displays the following SQL commands and results:

```

mysql> select * from Customer;
+-----+-----+-----+-----+-----+-----+-----+
| cid | name | phone | outDateTime | email | temperature | birthDate | a |
+-----+-----+-----+-----+-----+-----+-----+
| 2 | John Watson | +91 98761 22222 | 2022-01-08 10:39:52 | john.watson@example.com | 98.5 | 1990-08-08 | |
| 3 | Leo | +91 98761 54452 | 2022-01-09 10:39:52 | leo@example.com | 99.5 | 1990-08-08 | |
+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> select * from Customer;

```

A record inserted can now be seen in the output as:

The screenshot shows the Eclipse IDE with a terminal window running MySQL. The terminal displays the output of a `select * from Customer;` query, showing three rows of customer data. The Java file `DB.java` is also visible, showing a `public void createCustomer()` method that inserts a new customer into the database.

```

mysql> select * from Customer;
+----+-----+-----+-----+-----+-----+
| cid | name  | phone | outDateTime | email          | temperature | birthDate |
+----+-----+-----+-----+-----+-----+
| 32  | John  | +91 98761 22222 | 2022-01-08 10:39:52 | john.watson@example.com | 98.5 | 1990-08-08 |
| 32  | Leo   | +91 98761 54452 | 2022-01-08 10:39:52 | leo@example.com | 99.5 | 1990-08-08 |
| 32  | George | +91 99009 88088 | 2022-01-12 10:39:52 | george@example.com | 99.5 | 1986-04-04 |
+----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
db.createCustomer(customer);
//db.updateCustomer(customer);
//db.deleteCustomer(1);
//System.out.println();
//ArrayList<Customer> customers = db.getAllCustomers();
//customers.forEach( cRef -> System.out.println(cRef));
db.closeConnection();
  
```

## Step 2: Perform the Update operation

2.1 Open **Eclipse**. Go to the **DB.java** file and find the **updateCustomer()** method. Comment the SQL string and substitute it with the wildcard characters.

The screenshot shows the Eclipse IDE with the `DB.java` file open. The `updateCustomer()` method is visible, showing a SQL update statement that is commented out. The method uses wildcard characters to update the customer's name, phone, email, birthDate, age, inDateTime, and outDateTime.

```

//int result = statement.executeUpdate(sql);
//int result = preparedStatement.executeUpdate();
String message = result > 0 ? "Customer Created Successfully" : "Customer Not Created. Please Try Again";
System.out.println(TAG+message);
} catch (Exception e) {
    System.out.println("Exception Occurred: "+e);
}
}

public void updateCustomer(Customer customer) {
    try {
        //String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getEmail()+"', birthDate = '"+customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', outDateTime = '"+customer.getOutDateTime()+"' where cid = '"+customer.getCid()+'";
        String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getEmail()+"', birthDate = '"+customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', outDateTime = '"+customer.getOutDateTime()+"' where cid = '"+customer.getCid()+'";

        System.out.println("SQL is: "+sql);

        int result = statement.executeUpdate(sql);
        String message = result > 0 ? "Customer Updated Successfully" : "Customer Not Updated. Please Try Again";
        System.out.println(TAG+message);
    } catch (Exception e) {
        System.out.println("Exception Occurred: "+e);
    }
}
  
```



2.2 Write the lines of code from 124 to 141 in which we are creating the **preparedStatment** for updating the customer details

```

108     } catch (Exception e) {
109         System.out.println("Exception Occurred: "+e);
110     }
111 }
112
113
114 public void updateCustomer(Customer customer) {
115     try {
116
117         String sql = "update Customer set name = '"+customer.getName()+"', phone = '"+customer.getPhone()+"', email = '"+customer.getEmail()
118         +"', birthDate = '"+customer.getBirthDate()+"', age = '"+customer.getAge()+"', inDateTime = '"+customer.getInDateTime()+"', outDateTime = '"+customer.getO
119         +"' where cid = '"+customer.getCid()+'";
120
121         String sql = "update Customer set name = ?, phone = ?, email = ?, birthDate = ?, age = ?, inDateTime = ?, outDateTime = ?, temperature = ? where cid = ?"
122         System.out.println("SQL is: "+sql);
123
124         preparedStatement = connection.prepareStatement(sql);
125
126         preparedStatement.setString(1, customer.getName());
127         preparedStatement.setString(2, customer.getPhone());
128         preparedStatement.setString(3, customer.getEmail());
129         preparedStatement.setString(4, customer.getBirthDate());
130         preparedStatement.setInt(5, customer.getAge());
131         preparedStatement.setString(6, customer.getInDateTime());
132         preparedStatement.setString(7, customer.getOutDateTime());
133         preparedStatement.setFloat(8, customer.getTemperature());
134
135         preparedStatement.setInt(9, customer.getCid());
136
137         //int result = statement.executeUpdate(sql);
138         int result = preparedStatement.executeUpdate();
139         String message = result > 0 ? "Customer Updated Successfully" : "Customer Not Updated. Please Try Again";
140         System.out.println(TAG+message);
141     } catch (Exception e) {
142         System.out.println("Exception Occurred: "+e);
143     }
144 }
145

```

2.3 Go to the **App.java** file, uncomment the **updateCustomer** call, and comment on the **createCustomer** call. Make a few changes in customer details to update.

```

9  * Hello world!
10  *
11  */
12  public class App
13  {
14      public static void main( String[] args )
15      {
16          System.out.println( "Welcome to Customer Management System" );
17
18          Customer customer = new Customer();
19
20          //customer.setCid(3);
21          customer.setName("George G");
22          customer.setPhone("+91 99009 11111");
23          customer.setEmail("george.g@example.com");
24          customer.setBirthDate("1986-04-04");
25          customer.setAge(35);
26          customer.setInDateTime("2022-01-12 09:00:00");
27          customer.setOutDateTime("2022-01-12 10:30:00");
28          customer.setTemperature(98.8f);
29
30          System.out.println("Connecting to DB...");
31          DB db = new DB();
32          db.createConnection();
33

```

2.4 Run the code with changes. You can see the updated customer details.

The screenshot shows the Eclipse IDE with a Java file named `App.java` open. The code defines a `Customer` class and a `main` method. The `main` method creates a `Customer` object, sets its attributes, and updates it in a database. The console output shows the following messages:

```
<terminated> App [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
Welcome to Customer Management System
Connecting to DB...
[DB] Driver Loaded
[DB] Connection Created
SQL is: update Customer set name = ?, phone = ?, email = ?, birthDate = ?
[DB] Customer Updated Successfully
[DB] Connection Closed. Close Status: true
```

2.5 Open the terminal and run the **select** command. You can see a customer with **id = 3** has been updated.

The screenshot shows the Eclipse IDE with a terminal window open. The terminal displays the output of a `select` command executed in a MySQL database. The output shows a table with columns: `id`, `name`, `phone`, `email`, `birthDate`, `inDateTime`, `outDateTime`, and `temperature`. The table contains three rows of data. The third row, with `id = 3`, shows the updated customer details.

id	name	phone	email	birthDate	inDateTime	outDateTime	temperature
2	John Watson	+91 98761 22222	john.watson@example.com	1990-08-08	2022-01-08 10:39:52	2022-01-08 11:45:22	98.5
3	George	+91 99009 88088	george@example.com	1986-04-04	2022-01-12 09:00:00	2022-01-12 10:30:00	98.8
4	George	+91 99009 88088	george@example.com	1986-04-04	2022-01-12 10:39:52	2022-01-12 11:45:22	99.5

## Step 3: Perform the Delete operation

- 3.1 Use the **delete** operation, use the **executeUpdate()** function and print a message for the successful deletion of the record (line 151 to line 160)

```

130 preparedStatement.setString(4, customer.getName());
131 preparedStatement.setInt(5, customer.getAge());
132 preparedStatement.setString(6, customer.getInDateTime());
133 preparedStatement.setString(7, customer.getOutDateTime());
134 preparedStatement.setFloat(8, customer.getTemperature());
135
136 preparedStatement.setInt(9, customer.getCid());
137
138 //int result = statement.executeUpdate(sql);
139 int result = preparedStatement.executeUpdate();
140 String message = result > 0 ? "Customer Updated Successfully" : "Customer Not Updated. Please Try Again";
141 System.out.println(TAG+message);
142
143 } catch (Exception e) {
144     System.out.println("Exception Occurred: "+e);
145 }
146
147 }
148
149 public void deleteCustomer(int cid) {
150     try {
151         //String sql = "delete from Customer where cid = "+cid;
152         String sql = "delete from Customer where cid = ?";
153
154         preparedStatement = connection.prepareStatement(sql);
155         preparedStatement.setInt(1, cid);
156
157         //int result = statement.executeUpdate(sql);
158         int result = preparedStatement.executeUpdate();
159
160         String message = result > 0 ? "Customer Deleted Successfully" : "Customer Not Deleted. Please Try Again";
161         System.out.println(TAG+message);
162
163     } catch (Exception e) {
164         System.out.println("Exception Occurred: "+e);
165     }
166 }
167

```

- 3.2 Call the **deleteCustomer()** method, save the code, and then run it in the **App.java** file. In the output, as you can see the customer record is deleted, and it prints **Customer Deleted Successfully**.

```

16 System.out.println("Welcome to Customer Management System");
17
18 Customer customer = new Customer();
19
20 customer.setCid(4);
21 customer.setName("George G");
22 customer.setPhone("+91 99009 11111");
23 customer.setEmail("george.g@example.com");
24 customer.setBirthDate("1986-04-04");
25 customer.setAge(35);
26 customer.setInDateTime("2022-01-12 09:00:00");
27 customer.setOutDateTime("2022-01-12 10:30:00");
28 customer.setTemperature(98.8f);
29
30 System.out.println("Connecting to DB...");
31 DB db = new DB();
32 db.createConnection();
33
34 //db.createCustomer(customer);
35 //db.updateCustomer(customer);
36
37 db.deleteCustomer(4);
38
39 //System.out.println();
40
41 //ArrayList<Customer> customers = db.getAllCustomers();
42 //customers.forEach( cRef -> System.out.println(cRef));
43
44 db.closeConnection();
45
46 }

```

The screenshot shows the Eclipse IDE with the `App.java` file open. The code defines a `main` method that creates a `Customer` object, connects to a database, and performs several database operations. The `Console` window on the right shows the output of the program, including the message "Welcome to Customer Management System" and the status of database operations.

```

world!

class App

lic static void main( String[] args )

System.out.println( "Welcome to Customer Management System" );

Customer customer = new Customer();

customer.setCid(4);
customer.setName("George G");
customer.setPhone("+91 99889 11111");
customer.setEmail("george.g@example.com");
customer.setBirthDate("1986-04-04");
customer.setAge(35);
customer.setInDateTime("2022-01-12 09:00:00");
customer.setOutDateTime("2022-01-12 10:30:00");
customer.setTemperature(98.8f);

System.out.println("Connecting to DB...");
DB db = new DB();
db.createConnection();

//db.createCustomer(customer);
//db.updateCustomer(customer);

db.deleteCustomer(3);

//System.out.println();

//ArrayList<Customer> customers = db.getAllCustomers();
//customers.forEach( cRef -> System.out.println(cRef));

db.closeConnection();
    
```

Console Output:

```

<terminated> App [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot...
Welcome to Customer Management System
Connecting to DB...
[DB] Driver Loaded
[DB] Connection Created
[DB] Customer Deleted Successfully
[DB] Connection Closed. Close Status: true
    
```

## Step 4: Perform the getAllCustomers operation

### 4.1 Comment out the ResultSet in the DB.java file

The screenshot shows the Eclipse IDE with the `DB.java` file open. The `getAllCustomers` method is highlighted, showing the SQL query and the logic to fetch and return a list of `Customer` objects. The `ResultSet` object is commented out.

```

public ArrayList<Customer> getAllCustomers() {
    ArrayList<Customer> customers = new ArrayList<Customer>();

    try {
        String sql = "select * from Customer";
        //ResultSet set = statement.executeQuery(sql);

        while(set.next()) {
            Customer customer = new Customer();

            customer.setCid(set.getInt("cid"));
            customer.setName(set.getString(2));
            customer.setPhone(set.getString(3));
            customer.setEmail(set.getString(4));
            customer.setBirthDate(set.getString(5));
            customer.setAge(set.getInt(6));
            customer.setInDateTime(set.getString(7));
            customer.setOutDateTime(set.getString(8));
            customer.setTemperature(set.getFloat(9));

            customers.add(customer);
        }
    } catch (Exception e) {
        System.out.println("Exception Occurred: "+e);
    }

    return customers;
}
    
```

## 4.2 Create the **preparedStatement** and **ResultSet** object to call the **executeQuery()** function

```

171 public ArrayList<Customer> getAllCustomers() {
172
173     ArrayList<Customer> customers = new ArrayList<Customer>();
174
175     try {
176
177         String sql = "select * from Customer";
178         PreparedStatement = connection.prepareStatement(sql);
179         //ResultSet set = statement.executeQuery(sql);
180
181         ResultSet set = preparedStatement.executeQuery();
182
183         while(set.next()) {
184
185             Customer customer = new Customer();
186
187             customer.setCid(set.getInt("cid"));
188             customer.setName(set.getString(2));
189             customer.setPhone(set.getString(3));
190             customer.setEmail(set.getString(4));
191             customer.setBirthDate(set.getString(5));
192             customer.setAge(set.getInt(6));
193             customer.setInDateTime(set.getString(7));
194             customer.setOutDateTime(set.getString(8));
195             customer.setTemperature(set.getFloat(9));
196
197             customers.add(customer);
198
199         } catch (Exception e) {
200             System.out.println("Exception Occurred: "+e);
201         }
202
203         return customers;
204     }
205 }

```

## 4.3 Go to the **App.java** file. Uncomment the ArrayList and add a **forEach** loop to get all customer details.

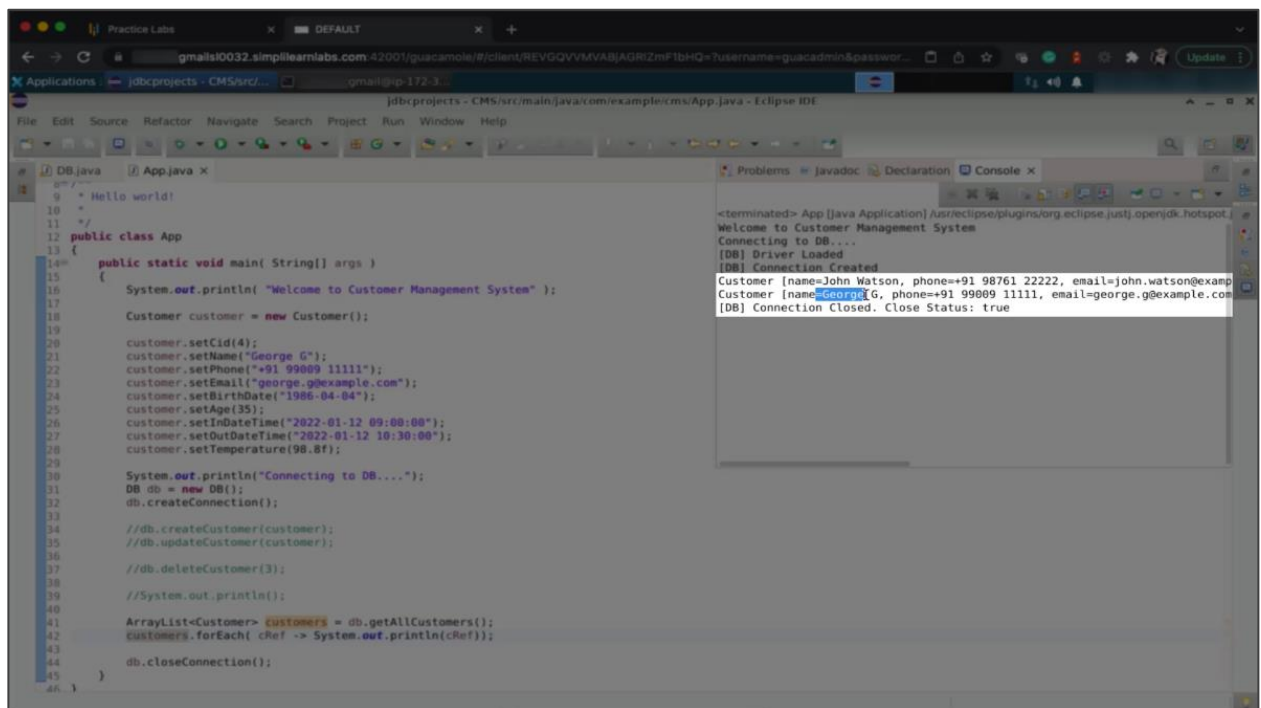
```

9  * Hello world!
10  *
11  */
12  public class App
13  {
14
15      public static void main( String[] args )
16      {
17          System.out.println( "Welcome to Customer Management System" );
18
19          Customer customer = new Customer();
20
21          customer.setCid(4);
22          customer.setName("George G");
23          customer.setPhone("+91 99009 11111");
24          customer.setEmail("george.g@example.com");
25          customer.setBirthDate("1986-04-04");
26          customer.setAge(35);
27          customer.setInDateTime("2022-01-12 09:00:00");
28          customer.setOutDateTime("2022-01-12 10:30:00");
29          customer.setTemperature(98.8f);
30
31          System.out.println("Connecting to DB...");
32          DB db = new DB();
33          db.createConnection();
34
35          //db.createCustomer(customer);
36          //db.updateCustomer(customer);
37          //db.deleteCustomer(3);
38          //System.out.println();
39
40          ArrayList<Customer> customers = db.getAllCustomers();
41          customers.forEach( cRef -> System.out.println(cRef));
42
43          db.closeConnection();
44      }
45  }

```



#### 4.4 Run the code, and you will see all customer details as output on the console



```

1  DB.java  2  App.java
9  * Hello world!
10
11  */
12  public class App
13  {
14      public static void main( String[] args )
15      {
16          System.out.println( "Welcome to Customer Management System" );
17
18          Customer customer = new Customer();
19
20          customer.setCid(4);
21          customer.setName("George G");
22          customer.setPhone("+91 99889 11111");
23          customer.setEmail("george.g@example.com");
24          customer.setBirthDate("1986-04-04");
25          customer.setAge(35);
26          customer.setInDateTime("2022-01-12 09:00:00");
27          customer.setOutDateTime("2022-01-12 10:30:00");
28          customer.setTemperature(98.8f);
29
30          System.out.println("Connecting to DB...");
31          DB db = new DB();
32          db.createConnection();
33
34          //db.createCustomer(customer);
35          //db.updateCustomer(customer);
36          //db.deleteCustomer(3);
37
38          //System.out.println();
39
40          ArrayList<Customer> customers = db.getAllCustomers();
41          customers.forEach( cRef -> System.out.println(cRef));
42
43          db.closeConnection();
44      }
45  }
46

```

```

<terminated> App [Java Application] /usr/eclipse/plugins/org.eclipse.justi.openjdk.hotspot.jre.full/bin/linux64/java
Welcome to Customer Management System
Connecting to DB...
[DB] Driver Loaded
[DB] Connection Created
Customer [name=John Watson, phone=+91 98761 22222, email=john.watson@example.com]
Customer [name=George G, phone=+91 99889 11111, email=george.g@example.com]
[DB] Connection Closed. Close Status: true

```

By following these steps, you have successfully implemented **PreparedStatement** API with JDBC for writing CRUD operations for managing database records efficiently and securely.