Lesson 01 Demo 05

Executing Batch Processing

Objective: To utilize Batch Processing and perform multiple SQL statements in a batch for efficient execution and management of large datasets. This approach minimizes database round-trips and enhances performance.

Tool required: Eclipse IDE

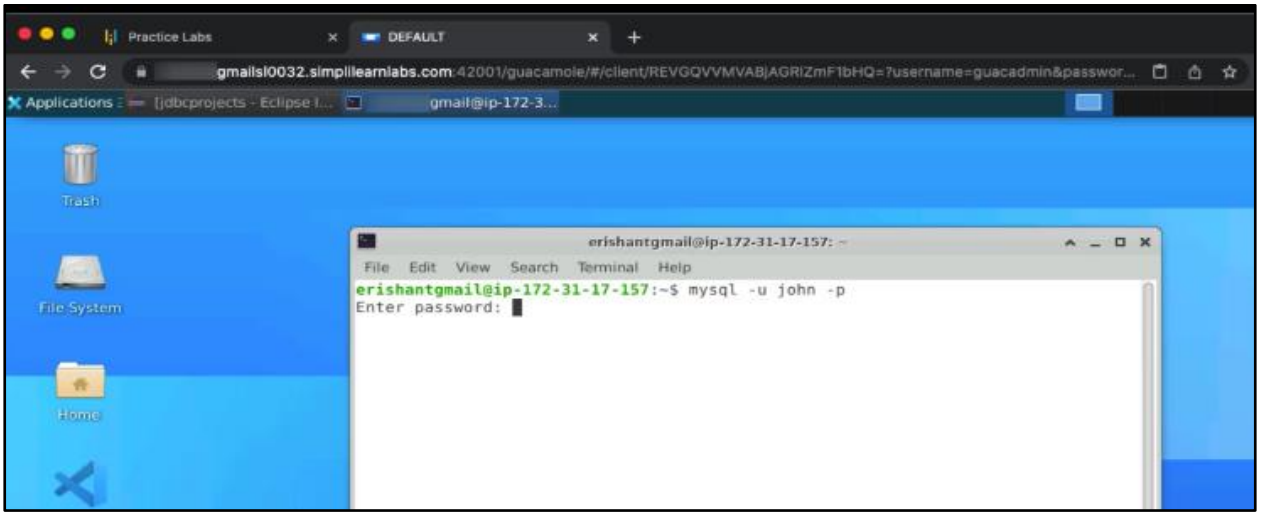
Prerequisites: None

Steps to be followed:

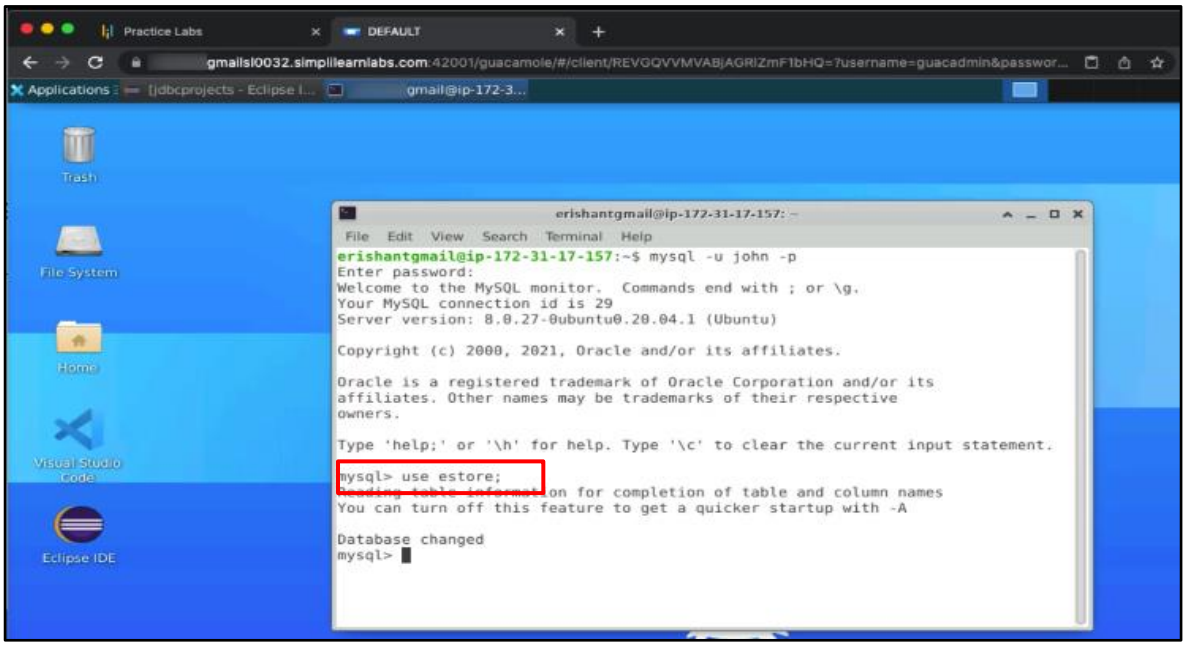
1. Create a table for employee details
2. Create a method for batch execution
3. Use a prepared statement
4. Use mixed SQL statements in a batch

Step 1: Create a table for employee details

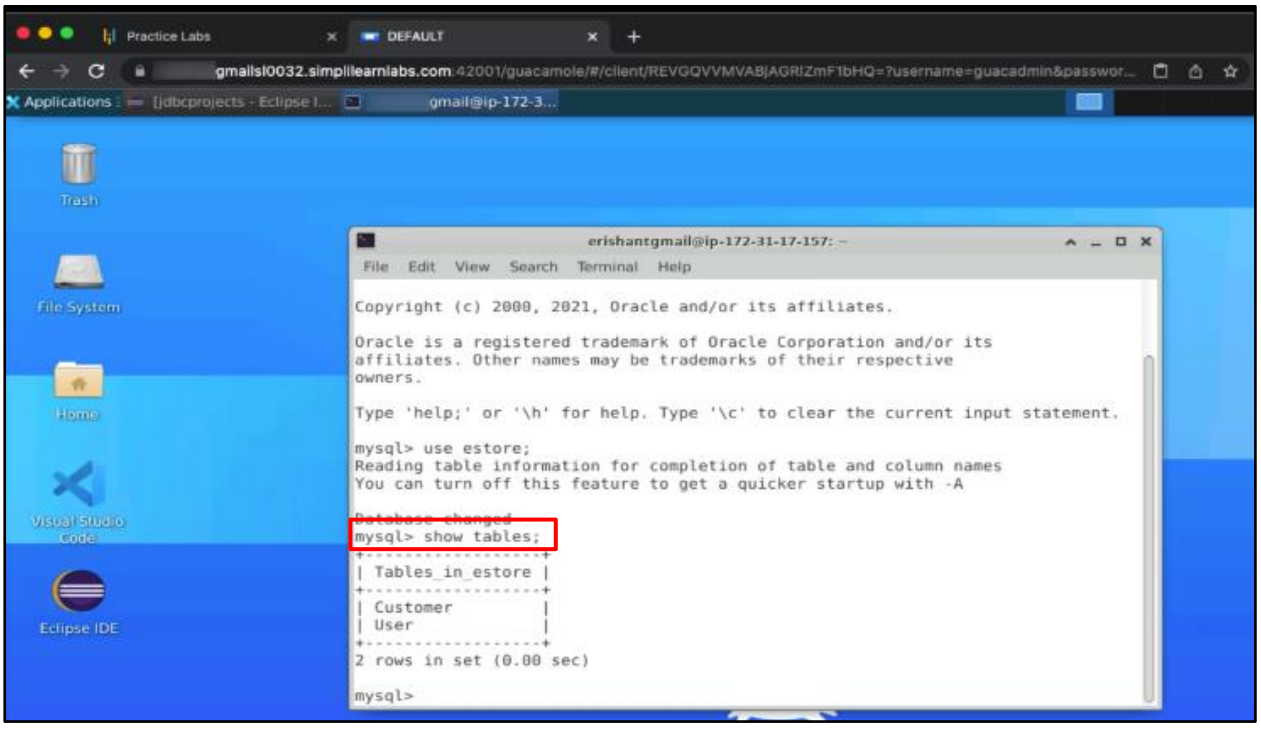
1.1 Open the terminal window and log in to MySQL using the following command:
mysql -u john -p



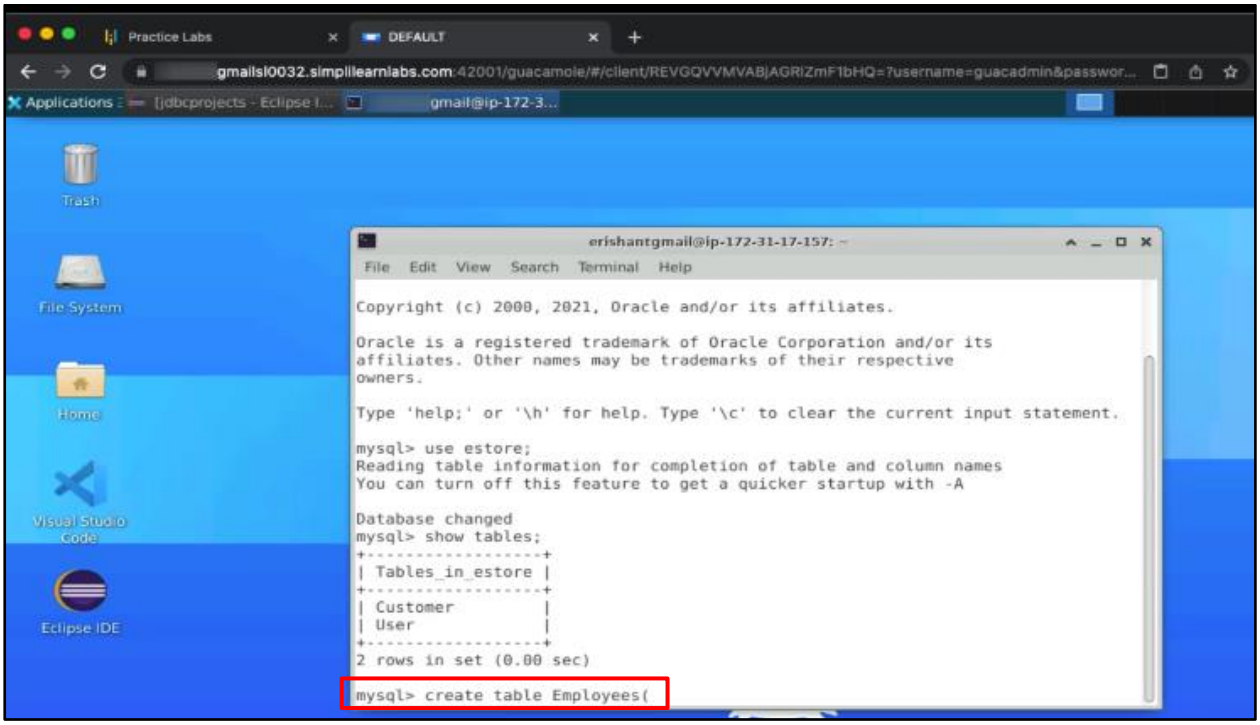
1.2 Enter the **use estore;** command to change the database



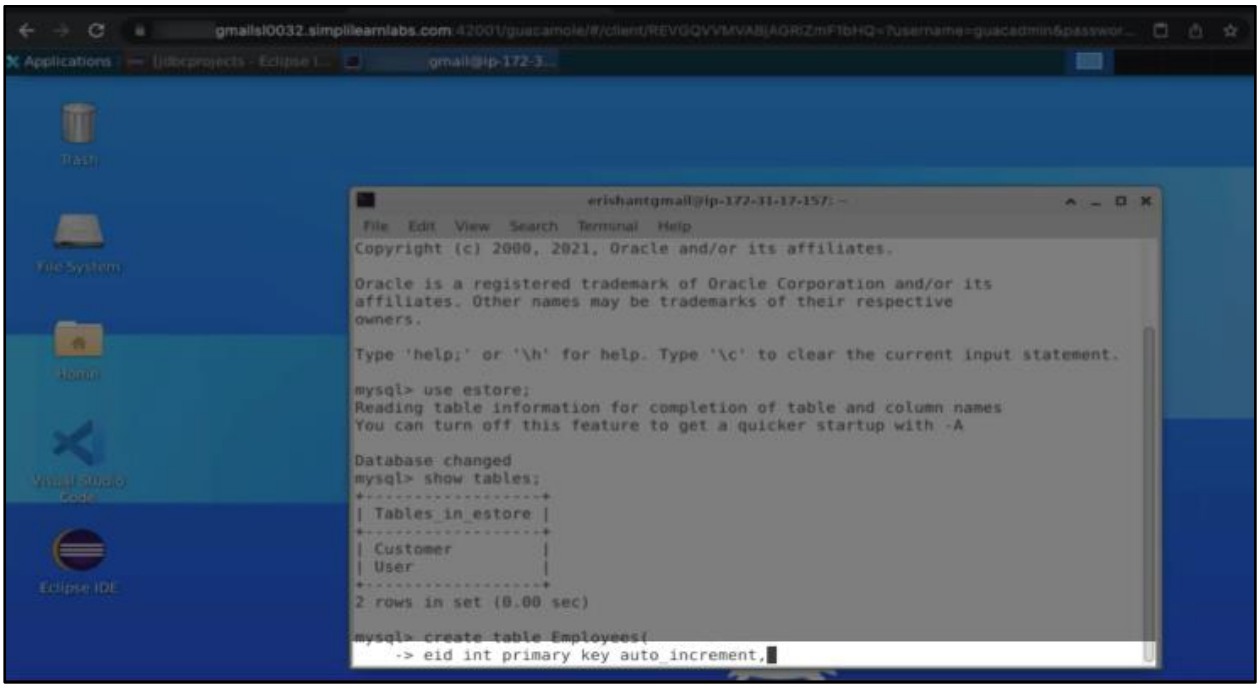
1.3 Run **show tables;** command to list tables in the **estore** database



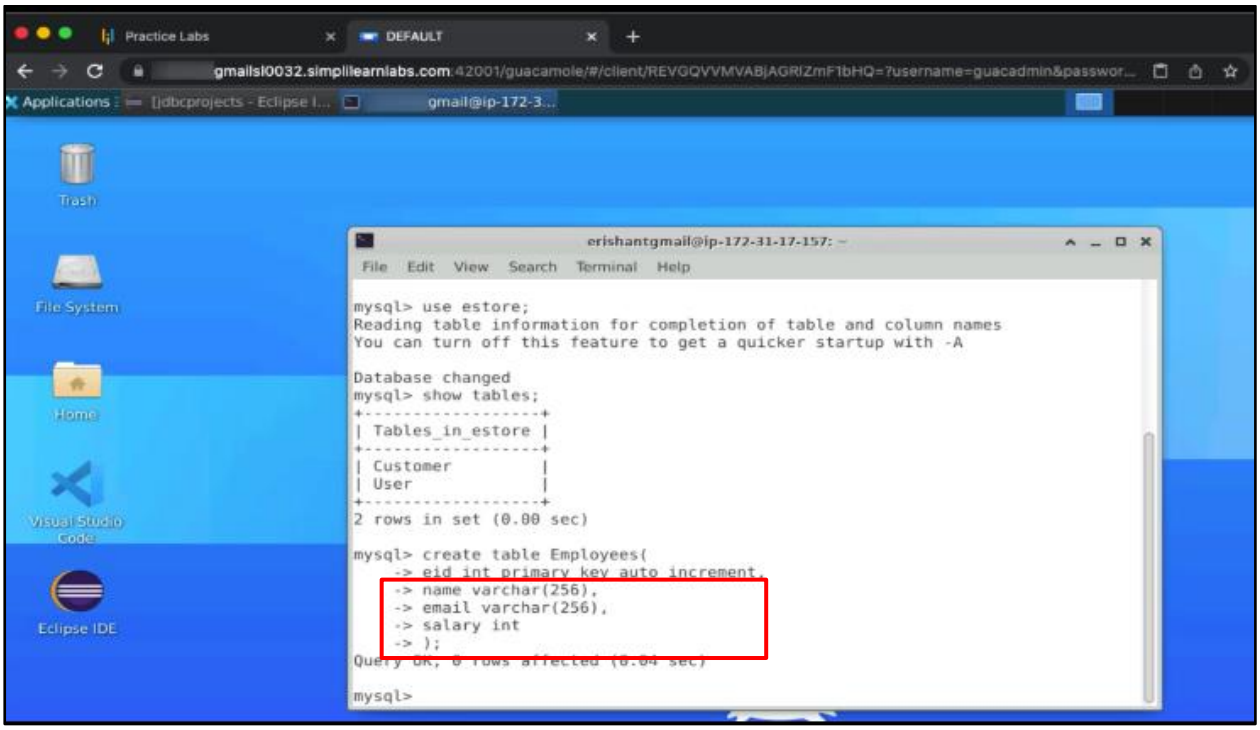
1.4 Run **create table Employees()** command to create a new table in the **estore** database



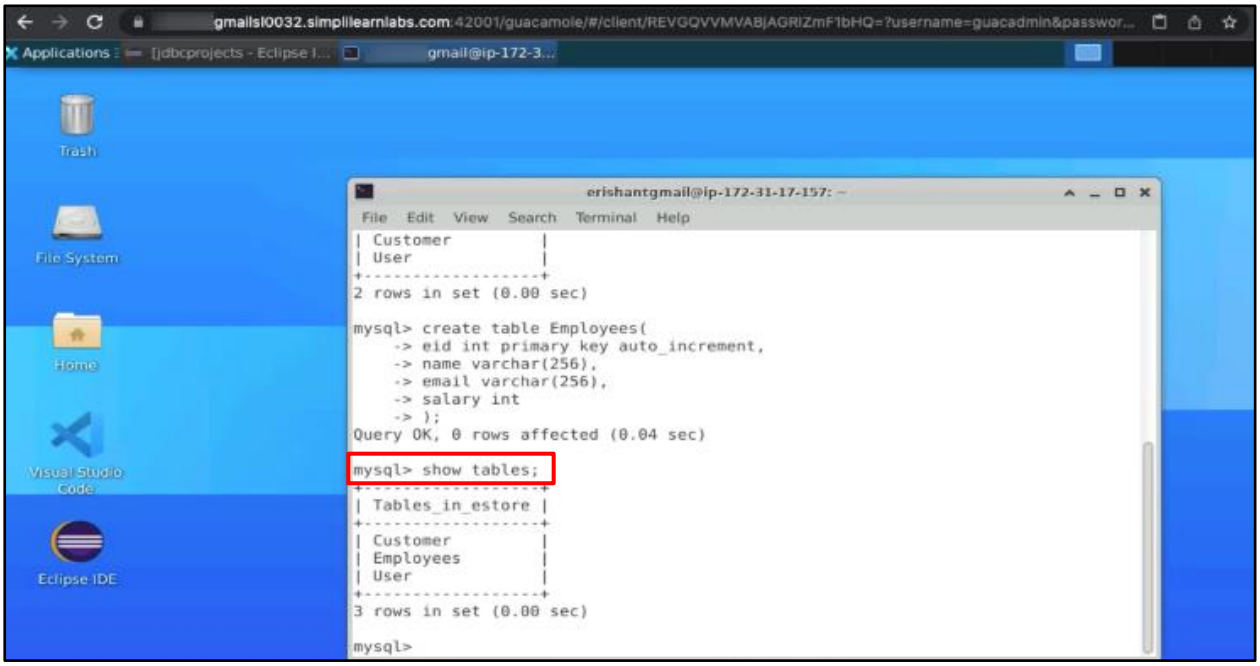
1.5 Enter the attribute **eid int primary key auto_increment,** to create a column for employee id



1.6 Add other attributes **name varchar (256)**, **email varchar (256)**, and **salary int** to the table

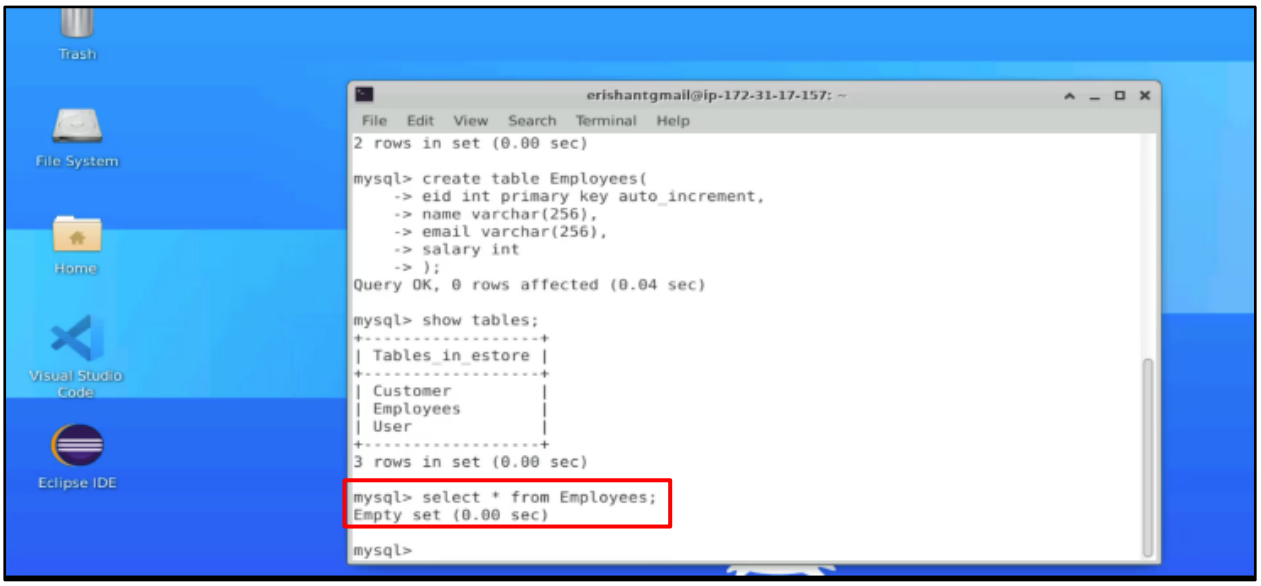


1.7 Enter **show tables;** command to check the table is created



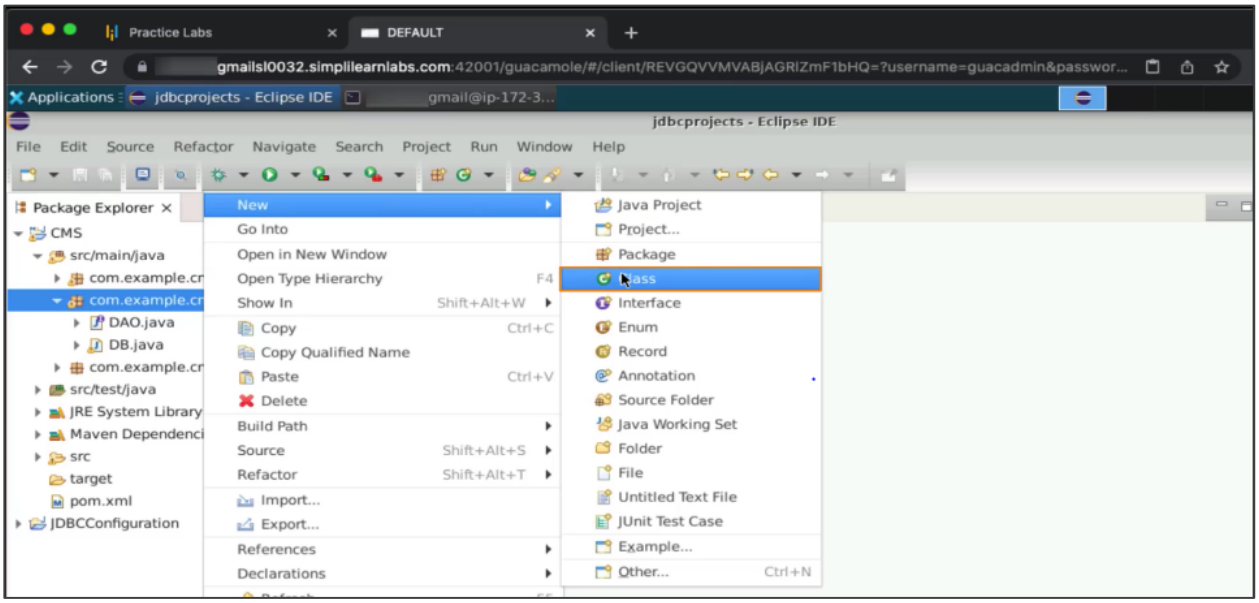
1.8 Run the **select** command to check the data in the table created. For now, the table is empty.

select * from Employees;

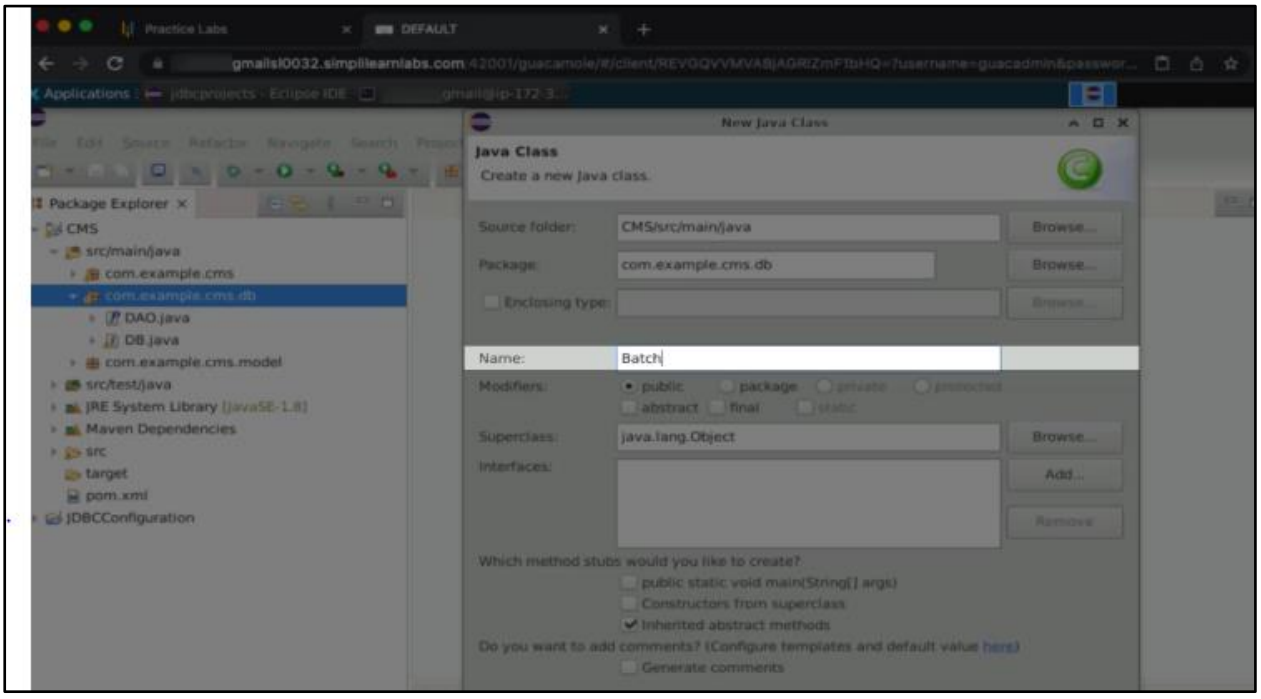


Step 2: Create a method for batch execution

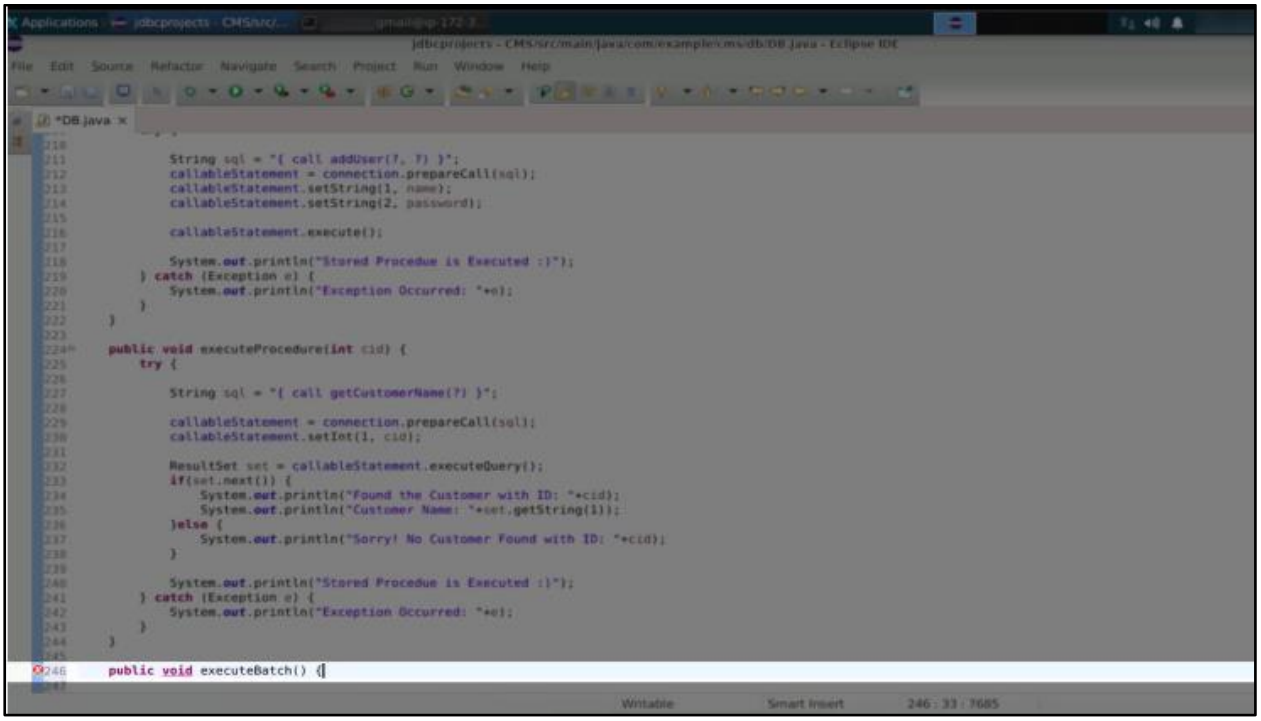
2.1 Create a new class in the package DB, open **Eclipse IDE**, go to **CMS** in the folder section **> src > com.example.cms**, then right-click, select **New**, and click on **Class**



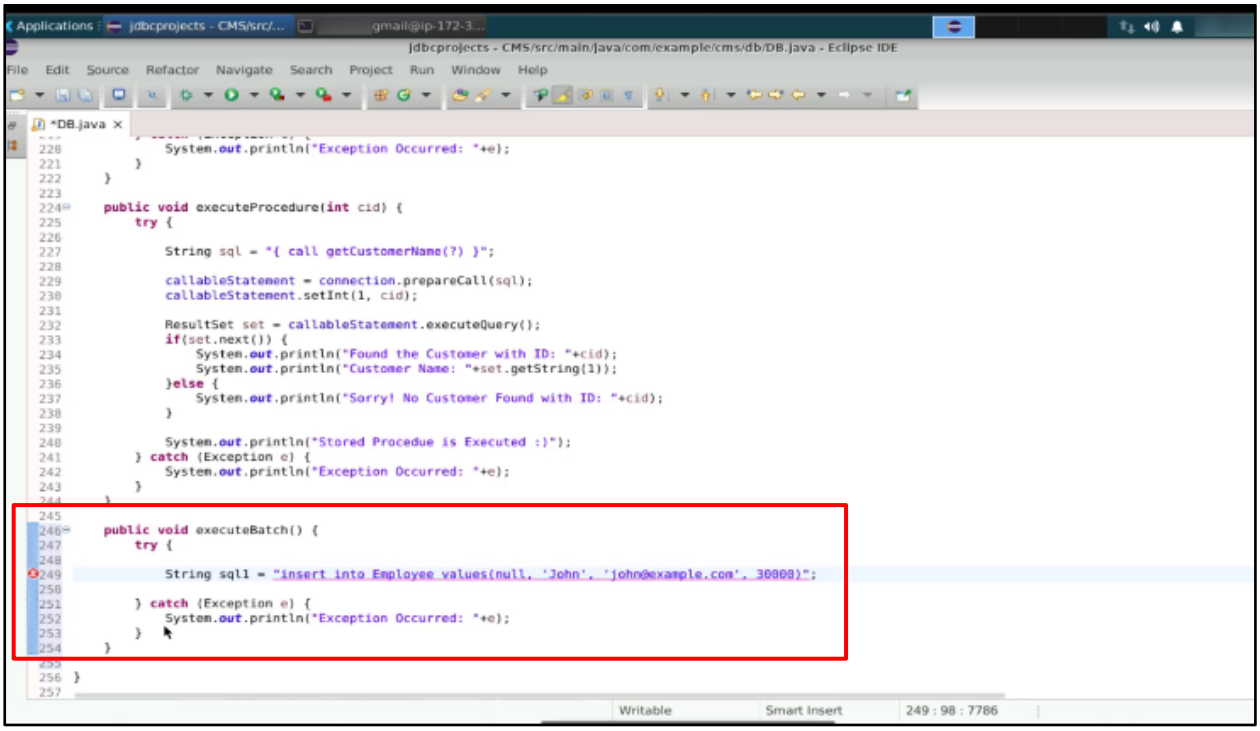
2.2 Name the class **Batch**



2.3 Create a new method **executeBatch()** for the batch execution process



2.4 Create a **try-catch** block with multiple SQL statements




```
            ResultSet set = callableStatement.executeQuery();
            if(set.next()) {
                System.out.println("Found the Customer with ID: "+cid);
                System.out.println("Customer Name: "+set.getString(1));
            }else {
                System.out.println("Sorry! No Customer Found with ID: "+cid);
            }

            System.out.println("Stored Procedue is Executed :)");
        } catch (Exception e) {
            System.out.println("Exception Occurred: "+e);
        }
    }

    public void executeBatch() {
        try {

            String sql1 = "insert into Employee values(null, 'John', 'john@example.com', 30000)";
            String sql2 = "insert into Employee values(null, 'Fionna', 'fionna@example.com', 34000)";
            String sql3 = "insert into Employee values(null, 'Leo', 'leo@example.com', 40000)";
            String sql4 = "insert into Employee values(null, 'Mike', 'mike@example.com', 50000)";
            String sql5 = "insert into Employee values(null, 'Kim', 'kim@example.com', 65000)";

            statement.addBatch(sql1);

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

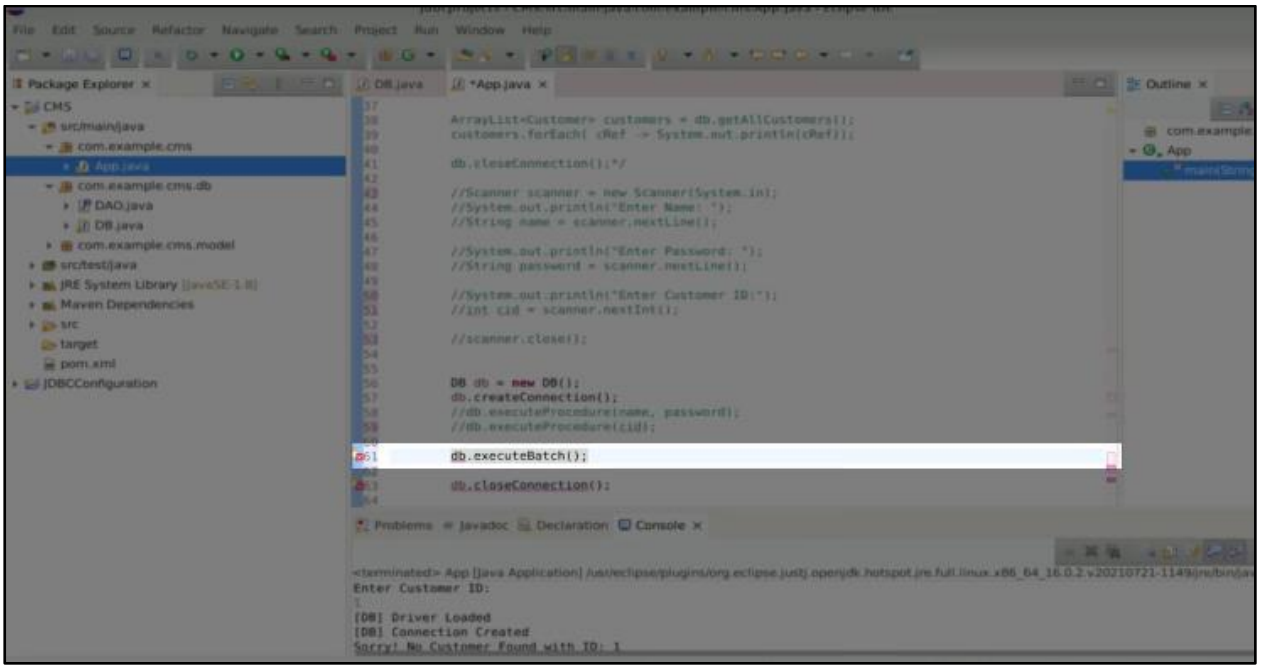
2.5 Execute a statement called **addbatch()** to create multiple SQL commands easily

```
        String sql = "{ call getCustomerName(?) }";

        callableStatement = connection.prepareCall(sql);
        callableStatement.setInt(1, cid);

        ResultSet set = callableStatement.executeQuery();
        if(set.next()) {
            System.out.println("Found the Customer with ID: "+cid);
            System.out.println("Customer Name: "+set.getString(1));
        }else {
            System.out.println("Sorry! No Customer Found with ID: "+cid);
        }

        System.out.println("Stored Procedue is Executed :)");
    } catch (Exception e) {
        System.out.println("Exception Occurred: "+e);
    }
}

public void executeBatch() {
    try {

        String sql1 = "insert into Employee values(null, 'John', 'john@example.com', 30000)";
        String sql2 = "insert into Employee values(null, 'Fionna', 'fionna@example.com', 34000)";
        String sql3 = "insert into Employee values(null, 'Leo', 'leo@example.com', 40000)";
        String sql4 = "insert into Employee values(null, 'Mike', 'mike@example.com', 50000)";
        String sql5 = "insert into Employee values(null, 'Kim', 'kim@example.com', 65000)";

        statement.addBatch(sql1);
        statement.addBatch(sql2);
        statement.addBatch(sql3);
        statement.addBatch(sql4);
        statement.addBatch(sql5);

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

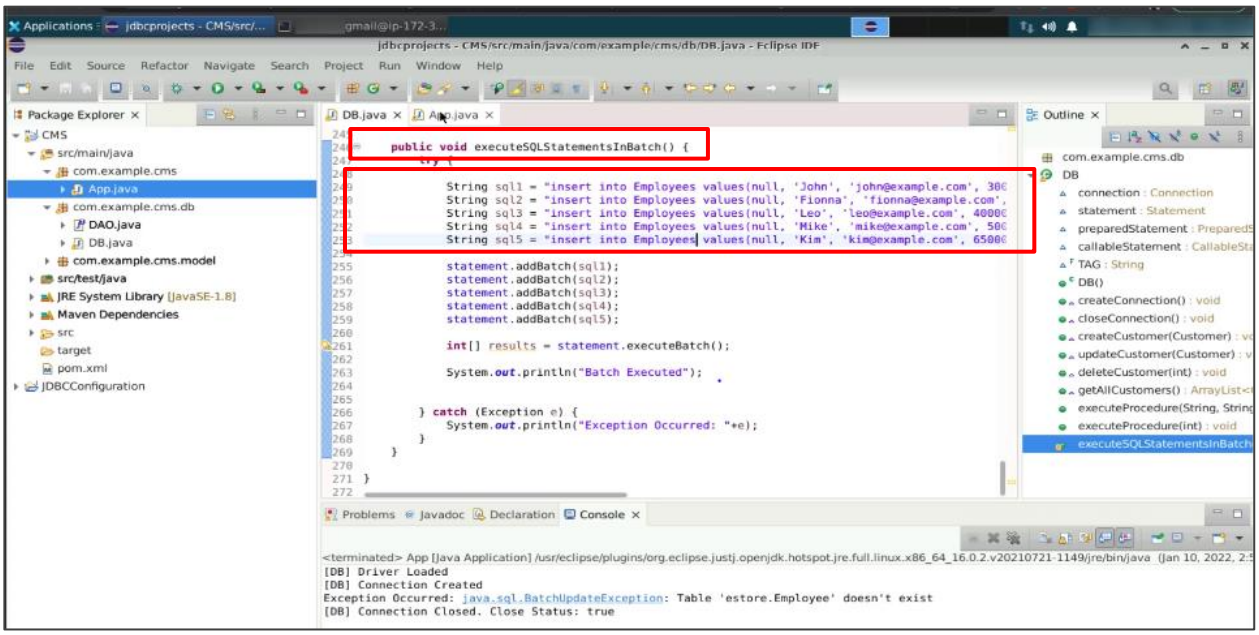
2.6 Enter **executeBatch();** statement, which will submit a batch of commands to the database for execution

```
        System.out.println("Stored Procedue is Executed :)");
    } catch (Exception e) {
        System.out.println("Exception Occurred: "+e);
    }
}

public void executeBatch() {
    try {

        String sql1 = "insert into Employee values(null, 'John', 'john@example.com', 30000)";
        String sql2 = "insert into Employee values(null, 'Fionna', 'fionna@example.com', 34000)";
        String sql3 = "insert into Employee values(null, 'Leo', 'leo@example.com', 40000)";
        String sql4 = "insert into Employee values(null, 'Mike', 'mike@example.com', 50000)";
        String sql5 = "insert into Employee values(null, 'Kim', 'kim@example.com', 65000)";

        statement.addBatch(sql1);
        statement.addBatch(sql2);
        statement.addBatch(sql3);
        statement.addBatch(sql4);
        statement.addBatch(sql5);

        int[] reults = statement.executeBatch();

    } catch (Exception e) {
```

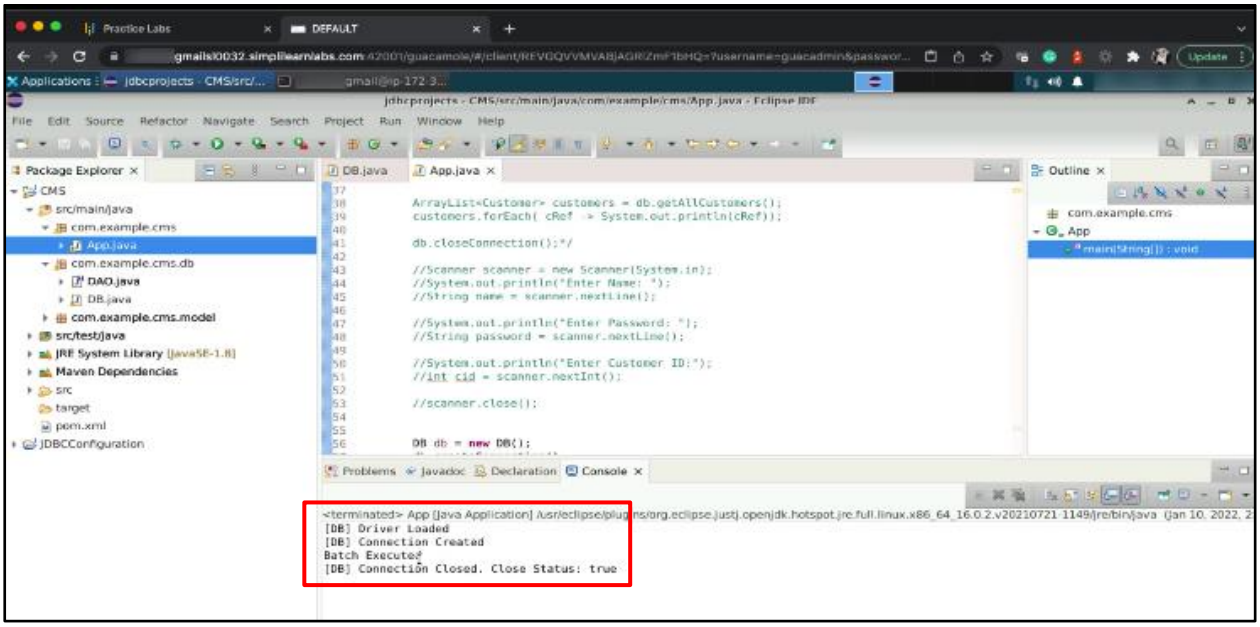
2.7 Go to **App.java** file comment statements on lines 43, 50, 51, 53, and 59, then **add db.executeBatch()** for SQL batch execution



2.8 Change the method name to **executeSQLStatmentsInBatch()** in both files to avoid confusion with other methods. Also, change **Employee** to **Employees** as the database name in the **DB.java** file

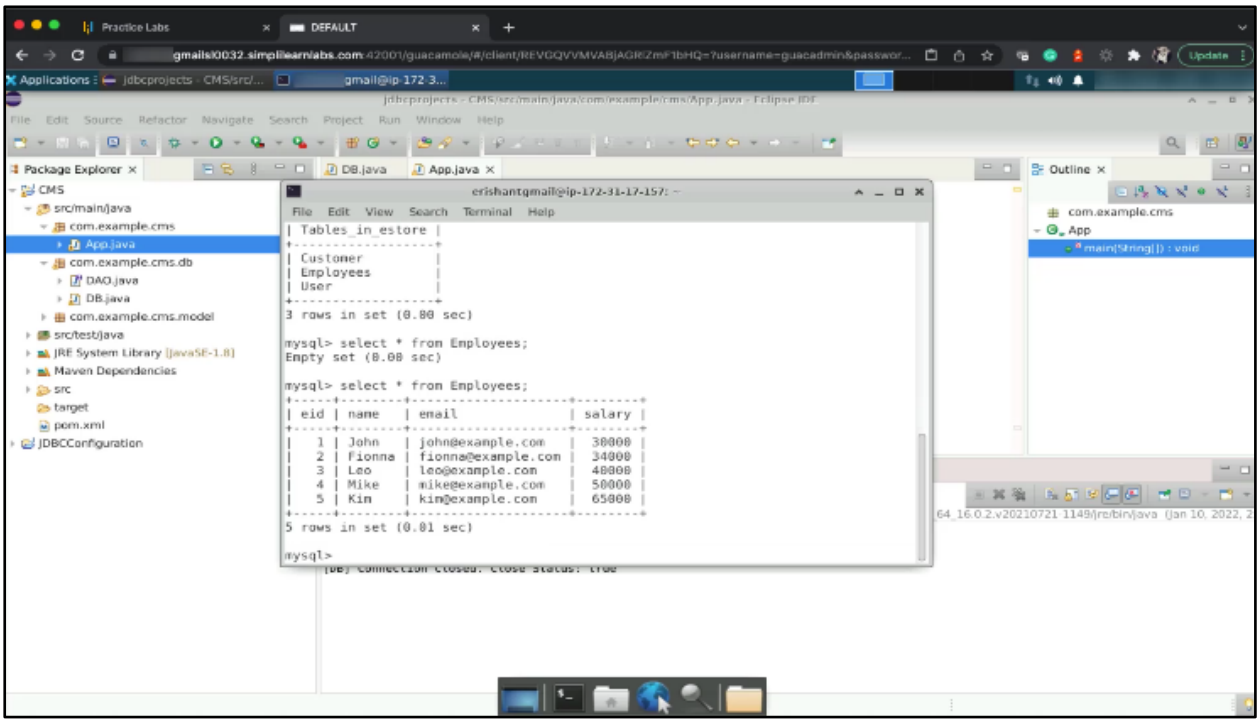


2.9 Save and run the code. You can see **Batch Executed** as output in the console.



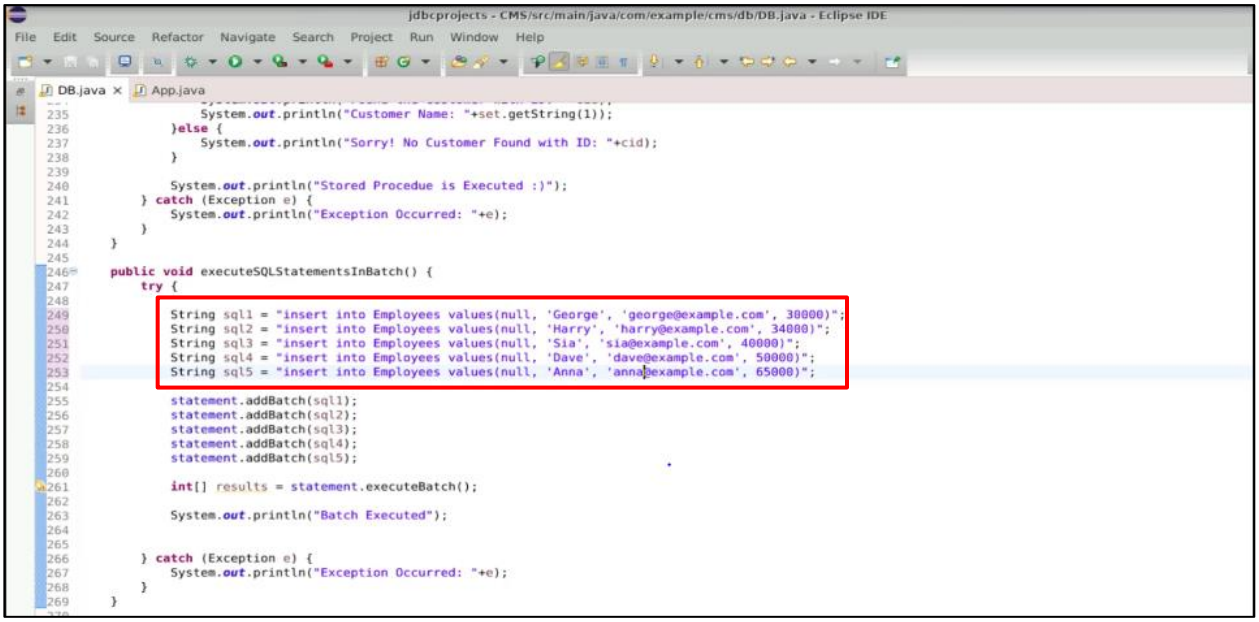
2.10 Rerun the select command in the terminal. You will get the list of employees inserted.

select * from Employees

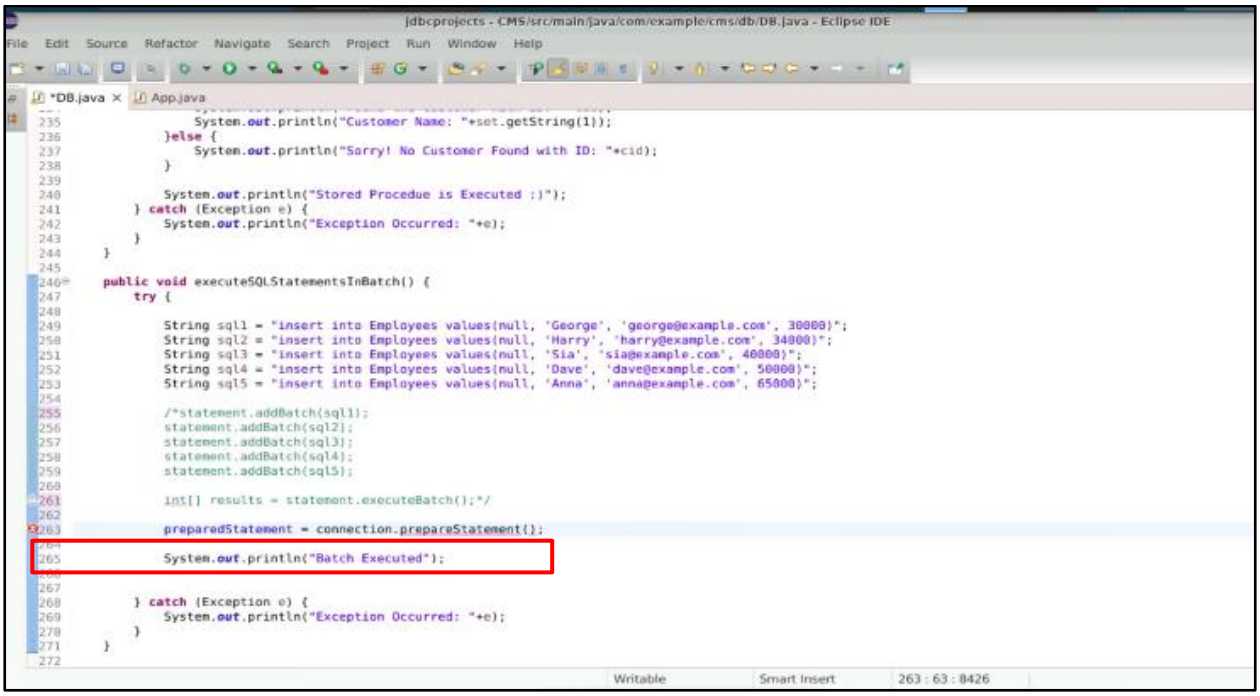


Step 3: Use a prepared statement

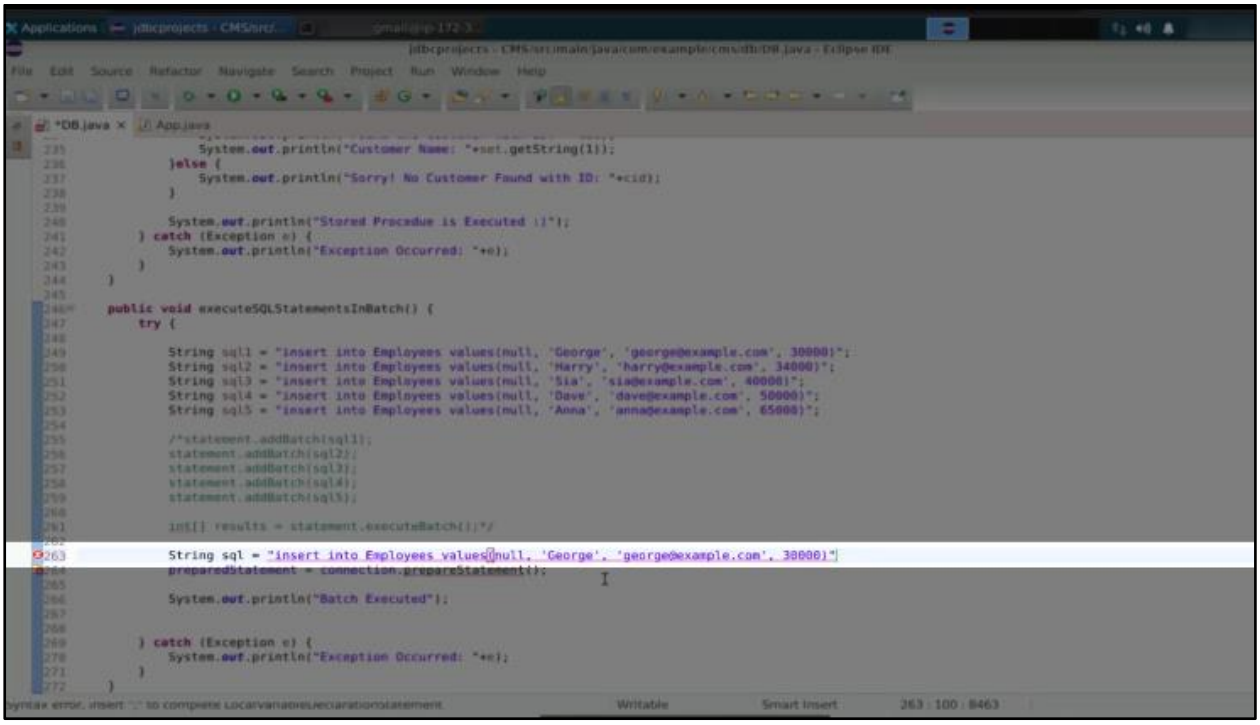
3.1 Change the details of the employees with new employee details in the **DB.java** file



3.2 Comment all the **addbatch()** statements and create a preparedStatement API:
preparedStatement = connection.preparedStatement();



3.3 Add a SQL statement for preparedStatement API to insert the data in the database



3.4 Use the **addBatch()** function to add a batch for the preparedStatement to enter the details of the employee's name **George**

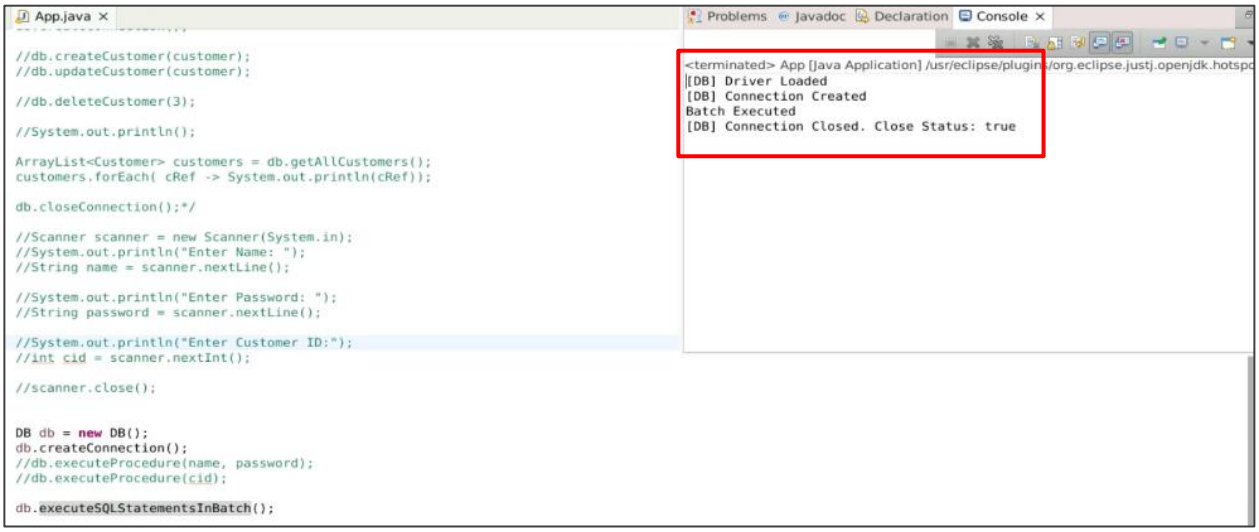
```
244     }
245
246     public void executeSQLStatementsInBatch() {
247         try {
248
249             String sql1 = "insert into Employees values(null, 'George', 'george@example.com', 30000)";
250             String sql2 = "insert into Employees values(null, 'Harry', 'harry@example.com', 34000)";
251             String sql3 = "insert into Employees values(null, 'Sia', 'sia@example.com', 40000)";
252             String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
253             String sql5 = "insert into Employees values(null, 'Anna', 'anna@example.com', 65000)";
254
255             /*statement.addBatch(sql1);
256             statement.addBatch(sql2);
257             statement.addBatch(sql3);
258             statement.addBatch(sql4);
259             statement.addBatch(sql5);
260
261             int[] results = statement.executeBatch();*/
262
263             String sql = "insert into Employees values(null, ?, ?, ?)";
264             preparedStatement = connection.prepareStatement(sql);
265
266             preparedStatement.setString(1, "George");
267             preparedStatement.setString(2, "george@example.com");
268             preparedStatement.setInt(3, 30000);
269             preparedStatement.addBatch();
270
271
272
273             System.out.println("Batch Executed");
274
275
276         } catch (Exception e) {
                                                    Writable    Smart Insert    269 : 42 : 8670
```

3.5 Repeat the above step for other records, and at the end, add the **preparedStatement** to execute the batch as:

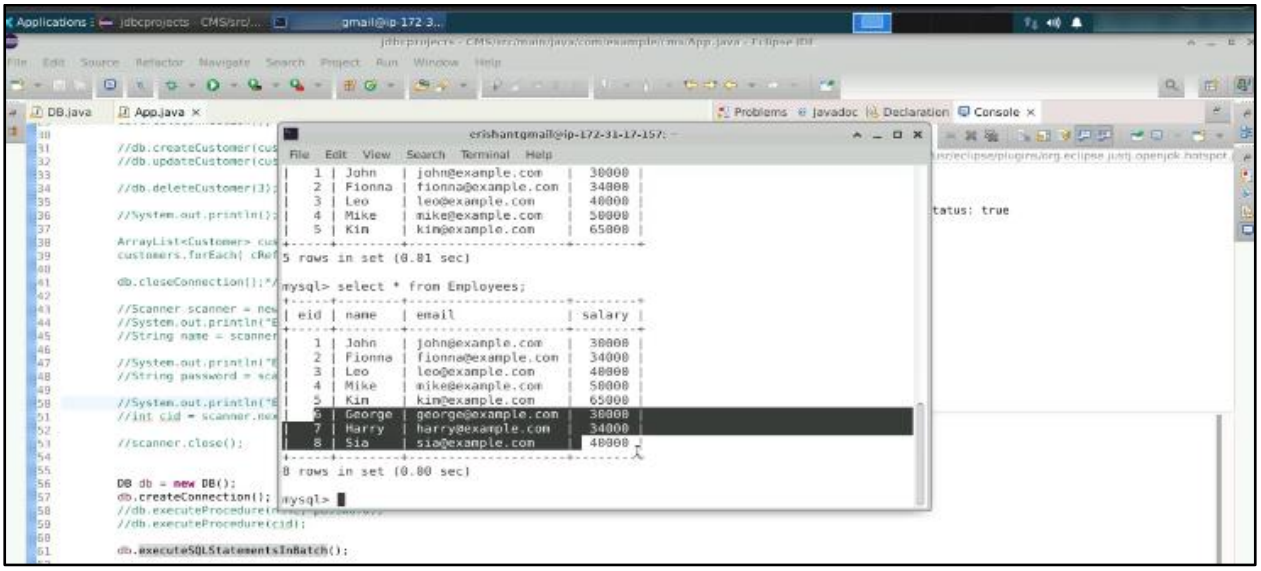
int[] = results = preparedStatement.executeBatch()

```
Applications : jdbcprojects - CMS/src/...    gmail@ip-172-3...
jdbcprojects - CMS/src/main/java/com/example/cms/db/DB.java - Eclipse IDE
File  Edit  Source  Refactor  Navigate  Search  Project  Run  Window  Help
DB.java ×  App.java
250             String sql2 = "insert into Employees values(null, 'Harry', 'harry@example.com', 34000)";
251             String sql3 = "insert into Employees values(null, 'Sia', 'sia@example.com', 40000)";
252             String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
253             String sql5 = "insert into Employees values(null, 'Anna', 'anna@example.com', 65000)";
254
255             /*statement.addBatch(sql1);
256             statement.addBatch(sql2);
257             statement.addBatch(sql3);
258             statement.addBatch(sql4);
259             statement.addBatch(sql5);
260
261             int[] results = statement.executeBatch();*/
262
263             String sql = "insert into Employees values(null, ?, ?, ?)";
264             preparedStatement = connection.prepareStatement(sql);
265
266             preparedStatement.setString(1, "George");
267             preparedStatement.setString(2, "george@example.com");
268             preparedStatement.setInt(3, 30000);
269             preparedStatement.addBatch();
270
271             preparedStatement.setString(1, "Harry");
272             preparedStatement.setString(2, "harry@example.com");
273             preparedStatement.setInt(3, 34000);
274             preparedStatement.addBatch();
275
276             preparedStatement.setString(1, "Sia");
277             preparedStatement.setString(2, "sia@example.com");
278             preparedStatement.setInt(3, 40000);
279             preparedStatement.addBatch();
280
281             int[] results = preparedStatement.executeBatch();
282
283             System.out.println("Batch Executed");
284
285
286         } catch (Exception e) {
287             System.out.println("Exception Occurred: "+e);
                                                    Writable    Smart Insert    281 : 29 : 9042
```

3.6 Go back to the **App.java** file and execute the code. You will see **Batch Executed** as the output without any error.



3.7 Go to the terminal and run the select command. The three new employee records get inserted



Step 4: Using mixed SQL statements in a batch

4.1 Mark the **preparedStatement** part as a comment

App.java

```
String sql2 = "insert into Employees values(null, 'Harry', 'harry@example.com', 34000)";
String sql3 = "insert into Employees values(null, 'Sia', 'sia@example.com', 40000)";
String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
String sql5 = "insert into Employees values(null, 'Anna', 'anna@example.com', 65000)";

/*statement.addBatch(sql1);
statement.addBatch(sql2);
statement.addBatch(sql3);
statement.addBatch(sql4);
statement.addBatch(sql5);

int[] results = statement.executeBatch();*/

/*String sql = "insert into Employees values(null, ?, ?, ?)";
preparedStatement = connection.prepareStatement(sql);

preparedStatement.setString(1, "George");
preparedStatement.setString(2, "george@example.com");
preparedStatement.setInt(3, 30000);
preparedStatement.addBatch();

preparedStatement.setString(1, "Harry");
preparedStatement.setString(2, "harry@example.com");
preparedStatement.setInt(3, 34000);
preparedStatement.addBatch();

preparedStatement.setString(1, "Sia");
preparedStatement.setString(2, "sia@example.com");
preparedStatement.setInt(3, 40000);
preparedStatement.addBatch();

int[] results = preparedStatement.executeBatch();*/

System.out.println("Batch Executed");
```

4.2 Add one **insertion SQL** statement to insert the data

```
public void executeSQLStatementsInBatch() {
    try {

        /*(String sql1 = "insert into Employees values(null, 'George', 'george@example.com', 30000)";
        String sql2 = "insert into Employees values(null, 'Harry', 'harry@example.com', 34000)";
        String sql3 = "insert into Employees values(null, 'Sia', 'sia@example.com', 40000)";
        String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
        String sql5 = "insert into Employees values(null, 'Anna', 'anna@example.com', 65000)";*/

        String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";

        statement.addBatch(sql1);
        statement.addBatch(sql2);
        statement.addBatch(sql3);
        statement.addBatch(sql4);
        statement.addBatch(sql5);

        int[] results = statement.executeBatch();

        /*String sql = "insert into Employees values(null, ?, ?, ?)";
        preparedStatement = connection.prepareStatement(sql);

        preparedStatement.setString(1, "George");
        preparedStatement.setString(2, "george@example.com");
        preparedStatement.setInt(3, 30000);
        preparedStatement.addBatch();

        preparedStatement.setString(1, "Harry");
        preparedStatement.setString(2, "harry@example.com");
        preparedStatement.setInt(3, 34000);
        preparedStatement.addBatch();

        preparedStatement.setString(1, "Sia");
```

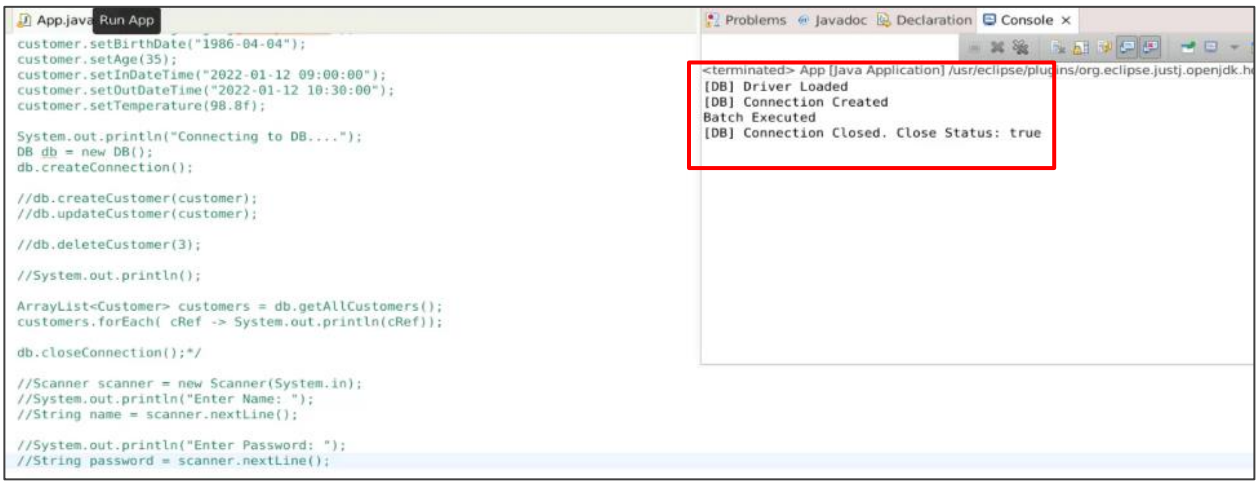
4.3 Add an SQL statement to update the **employee** table. Set the name as **Ron** for employee ID **2**

```
public void executeSQLStatementsInBatch() {
    try {

        /*(String sql1 = "insert into Employees values(null, 'George', 'george@example.com', 30000)";
        String sql2 = "insert into Employees values(null, 'Harry', 'harry@example.com', 34000)";
        String sql3 = "insert into Employees values(null, 'Sia', 'sia@example.com', 40000)";
        String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
        String sql5 = "insert into Employees values(null, 'Anna', 'anna@example.com', 65000)";*/

        String sql1 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
        String sql2 = "update Employees set name = 'Ron' where eid = 2";

        statement.addBatch(sql1);
        statement.addBatch(sql2);
        statement.addBatch(sql3);
        statement.addBatch(sql4);
        statement.addBatch(sql5);

        int[] results = statement.executeBatch();

        /*String sql = "insert into Employees values(null, ?, ?, ?)";
        preparedStatement = connection.prepareStatement(sql);

        preparedStatement.setString(1, "George");
        preparedStatement.setString(2, "george@example.com");
        preparedStatement.setInt(3, 30000);
        preparedStatement.addBatch();

        preparedStatement.setString(1, "Harry");
        preparedStatement.setString(2, "harry@example.com");
        preparedStatement.setInt(3, 34000);
```

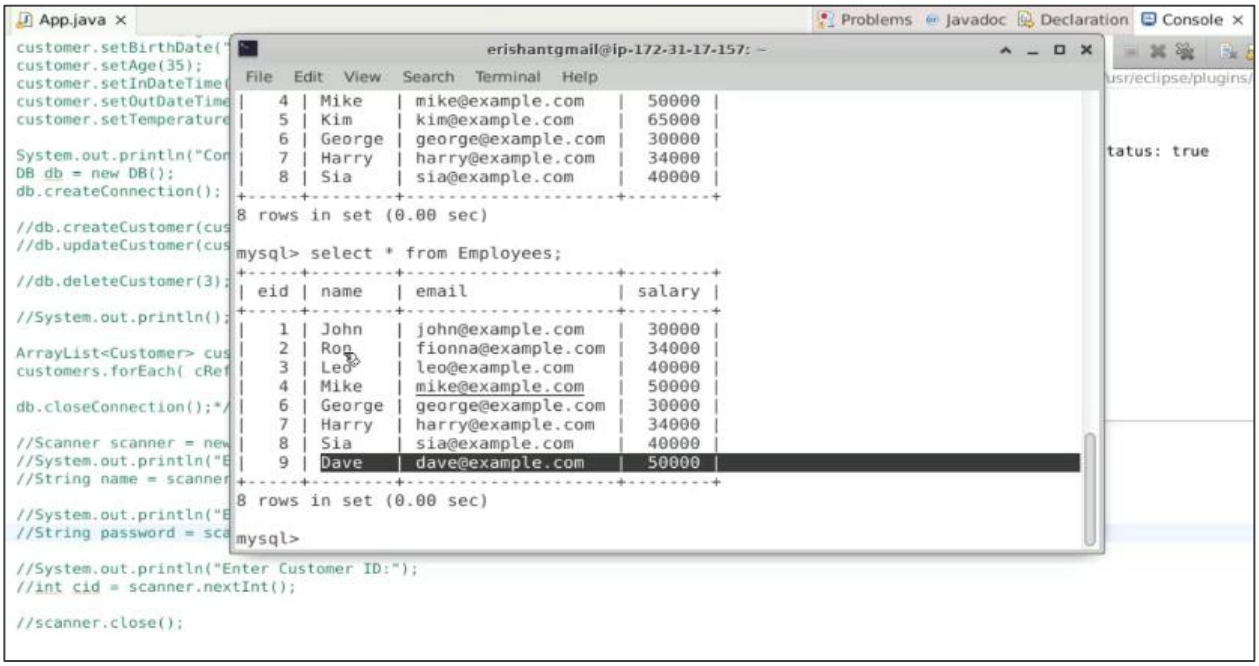
4.4 Add SQL **delete** statement to delete the employee with ID **5**

```
public void executeSQLStatementsInBatch() {
    try {

        /*(String sql1 = "insert into Employees values(null, 'George', 'george@example.com', 30000)";
        String sql2 = "insert into Employees values(null, 'Harry', 'harry@example.com', 34000)";
        String sql3 = "insert into Employees values(null, 'Sia', 'sia@example.com', 40000)";
        String sql4 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
        String sql5 = "insert into Employees values(null, 'Anna', 'anna@example.com', 65000)";*/

        String sql1 = "insert into Employees values(null, 'Dave', 'dave@example.com', 50000)";
        String sql2 = "update Employees set name = 'Ron' where eid = 2";
        String sql3 = "delete from Employees where eid = 5"

        statement.addBatch(sql1);
        statement.addBatch(sql2);
        statement.addBatch(sql3);
        statement.addBatch(sql4);
        statement.addBatch(sql5);

        int[] results = statement.executeBatch();

        /*String sql = "insert into Employees values(null, ?, ?, ?)";
        preparedStatement = connection.prepareStatement(sql);

        preparedStatement.setString(1, "George");
        preparedStatement.setString(2, "george@example.com");
        preparedStatement.setInt(3, 30000);
        preparedStatement.addBatch();

        preparedStatement.setString(1, "Harry");
        preparedStatement.setString(2, "harry@example.com");
        preparedStatement.setInt(3, 34000);
```

4.5 Save and run the code again. You will see the output as **Batch Executed** without any error.



4.6 Go to the terminal and run the **select** command. You can see all the SQL statements in the batch get executed successfully.



By following these steps, you have successfully executed Batch Processing and performed multiple SQL statements in a batch for efficient execution and management of large datasets. This approach minimizes database round-trips and enhances performance.