Experiment No. 8

Aim of the Experiment: Write a database application that uses any JDBC driver.

Objective:

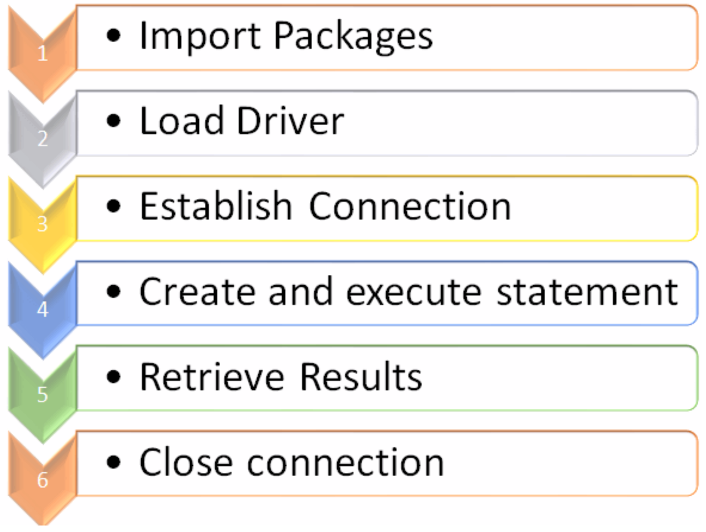
Create a database application using JDBC driver.

Course Outcome Addressed: CO4

Theory:

JDBC Connection Steps

There are 6 basic steps to connect with JDBC. They are enlisted in the below image:



1) Import Packages

First, we need to import the existing packages to use it in our Java program. Import will make sure that JDBC API classes are available for the program. We can then use the classes and subclasses of the packages.

Irrespective of the JDBC Driver, add the following import statement in the Java program.
import java.sql.*;

Import the other classes based on the functionality which you will use in the program. Download the appropriate Jar files for the database which you will use in the program.

.

JDBC API 4.0 mainly provides 2 important packages:

- java.sql
- javax.sql

(i) java.sql package

This package provides classes and interfaces to perform most of the JDBC functions like creating and executing SQL queries.

Classes/ Interfaces	Description
BLOB	It represents SQL Blob value in Java program
CallableStatement	It is used to execute SQL stored procedures
CLOB	It represents SQL Clob value in Java program
Connection	It creates a connection (session) with a specific Database
Date	It provides support for Date SQL type
Driver	It creates an instance of a Driver with Driver Manager
DriverManager	It provides basic service to manage a set of JDBC Drivers
ParameterMetaData	It is an object which can be used to get the information about the types and properties of each parameter in a PreparedStatement Object
PreparedStatement	It is used to create and execute a parameterized query in the Java program
ResultSet	It is used to access the result row-by-row
ResultSetMetaData	It is used to get the information about the types and properties of the columns in a ResultSet object
RowId	It represents the SQL ROWID value
Savepoint	It represents savepoint in transaction
SQLData	It is used to map the SQL User Defined Type (UDT) to a class in Java program
SQLXML	It represents SQL XML type
Statement	It is used to execute a static SQL statement
DriverPropertyInfo	It provides Driver properties to make a connection
SQLException	It provides information on database errors
SQLTimeoutException	It is a subclass of SQLException thrown when the timeout specified by the statement has expired

Classes/ Interfaces	Description
SQLWarning	It is an exception that provides information on database access warnings
Struct	It is a standard mapping in Java program for SQL structured type

(ii) javax.sql package
It is a JDBC extension API and provides server-side data access and processing in Java Program.

Classes/ Interfaces	**Description**
CommonDataSource	It is an interface that defines the methods which are common between DataSource, XADataSource and ConnectionPoolDataSource
ConnectionPoolDataSource	It is a factory for PooledConnection objects
DataSource	It is a factory for connections to the physical DataSource that the object represents
PooledConnection	It is used to manage Connection Pool
RowSet	It provides support to the JDBC API for Java beans Component Model
RowSetMetadata	It has the information about the columns in a RowSet object
ConnectionEvent	It provides information about the occurrence of connection-related events
ConnectionEventListener	It is used to register PooledConnection object events
RowSetEvent	It generates when an event occurs to a Rowset object
StatementEvent	It is sent to all StatementEventListeners which were registered with a PooledConnection generated

2) Load Driver

First, we should load/register the driver in the program before connecting to the Database. You need to register it only once per database in the program.

We can load the driver in the following 2 ways:

1. **Class.forName()**
2. **DriverManager.registerDriver()**

(i) Class.forName()
In this way, the driver's class file loads into the memory at runtime. It implicitly loads the driver. While loading, the driver will register with JDBC automatically.

DB Name	**JDBC Driver Name**
MySQL	com.mysql.jdbc.Driver
Oracle	oracle.jdbc.driver.OracleDriver
Microsoft SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver
MS Access	net.ucanaccess.jdbc.UcanaccessDriver
PostgreSQL	org.postgresql.Driver
IBM DB2	com.ibm.db2.jdbc.net.DB2Driver
Sybase	com.sybase.jdbcSybDriver
TeraData	com.teradata.jdbc.TeraDriver

Note: forName() method is valid only for JDK Compliant Virtual Machines.

(ii) DriverManager.registerDriver()
DriverManager is an inbuilt class that is available in the java.sql package. It acts as a mediator between Java application and database which you want to connect. Before you connect with the database, you need to register the driver with DriverManager. The main function of DriverManager is to load the driver class of the Database and create a connection with DB.

Public static void registerDriver(driver) – This method will register the driver with the Driver Manager. If the driver is already registered, then it won't take any action.

- It will throw **SQLException** if the database error occurs.
- It will throw **NullPointerException** if the driver is null.

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver())

DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver())

Like this, you can register the driver for your Database by passing it as a parameter.

3) Establish Connection

After loading the driver, the next step is to create and establish the connection. Once required, packages are imported and drivers are loaded and registered, then we can go for establishing a Database connection.

DriverManager class has the getConnection method, we will use this method to get the connection with Database. To call getConnection() method, we need to pass 3 parameters. The 3 parameters are string data type URL, a username, and a password to access the database.

The getConnection() method is an overloaded method. The 2 methods are:

- **getConnection(URL,username,password);** – It has 3 parameters URL, username, password.
- **getConnection(URL);** – It has only one parameter. URL has a username and password also.

The following table lists the JDBC connection strings for the different databases:

Database	**Connection String/DB URL**
MySQL	jdbc:mysql://HOST_NAME:PORT/DATABASE_NAME
Oracle	jdbc:oracle:thin:@HOST_NAME:PORT:SERVICE_NAME
Microsoft SQL Server	jdbc:sqlserver://HOST_NAME:PORT;DatabaseName=< DATABASE_NAME>
MS Access	jdbc:ucanaccess://DATABASE_PATH
PostgreSQL	jdbc:postgresql://HOST_NAME:PORT/DATABASE_NAME
IBM DB2	jdbc:db2://HOSTNAME:PORT/DATABASE_NAME
Sybase	jdbc:Sybase:Tds:HOSTNAME:PORT/DATABASE_NAME
TeraData	jdbc:teradata://HOSTNAME/database=< DATABASE_NAME>,tmode=ANSI,charset=UTF8

Example:
Connection con = DriverManager.getConnection(jdbc:oracle:thin:@localhost:1521:xe,System,Pass123@)

Here in this example,

- **thin** refers to the Driver type.
- **localhost** is where the Oracle database is running.
- **1521** is the port number to connect to DB.
- **xe** – SID
- **System** – User name to connect to the Oracle Database.
- **Pass123@** – Password

4) Create And Execute Statement

Once the connection has established, we can interact with the connected Database. First, we need to create the statement to perform the SQL query and then execute the statement.

(i) Create Statement
Now we will create the statement object that runs the query with the connected database. We use the createStatement method of the *Connection* class to create the query.
There are 3 statement interfaces are available in the java.sql package. These are explained below:
a) Statement
This interface is used to implement simple SQL statements with no parameter. It returns the ResultSet object.

Statement statemnt1 = conn.createStatement();

b) PreparedStatement
This PreparedStatement interface extends the Statement interface. So, it has more features than the Statement interface. It is used to implement parameterized and precompiled SQL statements. The performance of the application increases because it compiles the query only once.

It is easy to reuse this interface with a new parameter. It supports the IN parameter. Even we can use this statement without any parameter.

String select_query = “Select * from states where state_id = 1”;

PreparedStatement prpstmt = conn.prepareStatement(select_query);

c) CallableStatement
CallableStatement interface extends the PreparedStatement interface. So, it has more features than the PreparedStatement interface. It is used to implement a parameterized SQL statement that invokes procedure or function in the database. A stored procedure works like a method or function in a class. It supports the IN and OUT parameters.

The CallableStatement instance is created by calling the prepareCall method of the Connection object.

CallableStatementcallStmt = con.prepareCall("{call procedures(?,?)}");

(ii) Execute The Query
There are 4 important methods to execute the query in Statement interface. These are explained below:
- ResultSet executeQuery(String sql)
- int executeUpdate(String sql)
- boolean execute(String sql)
- int []executeBatch()

a) ResultSet executeQuery(String sql)
The executeQuery() method in Statement interface is used to execute the SQL query and retrieve the values from DB. It returns the ResultSet object. Normally, we will use this method for the SELECT query.

b) executeUpdate(String sql)
The executeUpdate() method is used to execute value specified queries like INSERT, UPDATE, DELETE (DML statements), or DDL statements that return nothing. Mostly, we will use this method for inserting and updating.

c) execute(String sql)
The execute() method is used to execute the SQL query. It returns **true** if it executes the SELECT query. And, it returns **false** if it executes INSERT or UPDATE query.
d) executeBatch()
This method is used to execute a batch of SQL queries to the Database and if all the queries get executed successfully, it returns an array of update counts. We will use this method to insert/update the bulk of records.

5) Retrieve Results

When we execute the queries using the executeQuery() method, the result will be stored in the ResultSet object. The returned ResultSet object will never be null even if there is no matching record in the table. ResultSet object is used to access the data retrieved from the Database.

ResultSet rs 1= statemnt1.executeQuery(QUERY));

We can use the executeQuery() method for the SELECT query. When someone tries to execute the insert/update query, it will throw SQLExecption with the message "**executeQuery method can not be used for update**".
A ResultSet object points to the current row in the Resultset. To iterate the data in the ResultSet object, call the next() method in a while loop. If there is no more record to read, it will return FALSE.

ResultSet can also be used to update data in DB. We can get the data from ResultSet using getter methods such as getInt(), getString(), getDate(). We need to pass the column index or column name as the parameter to get the values using Getter methods.

We will get to know more about the ResultSet in the next tutorial.

6) Close Connection

Finally, we are done with manipulating data in DB. Now we can close the JDBC connection. We need to make sure that we have closed the resource after we have used it. If we don't close them properly we may end up out of connections.

When we close the connection object, Statement and ResultSet objects will be closed automatically.

conn.close();

From Java 7 onwards, we can close the JDBC connections automatically using a try-catch block. JDBC connection should be opened in the parenthesis of the try block. Inside the try block, you can do the database connections normally as we do.

Once the execution exits the try block, it will automatically close the connection. In this case, we don't need to close the connection by calling conn.close method in the Java program.

```
try(Connection conn = DriverManager.getConnection(url, user, password))

{

    //database connection and operation

}
```

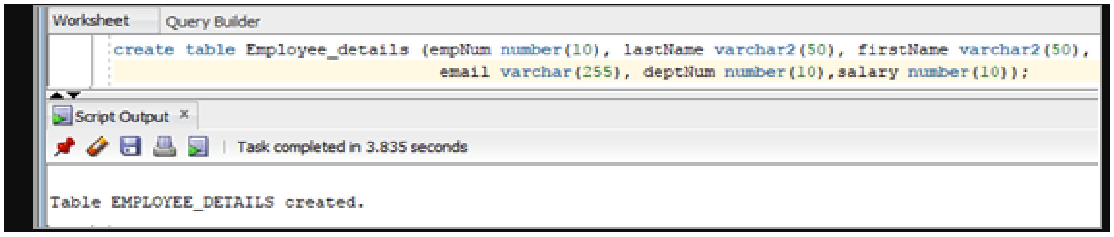
Java JDBC Connection Example
In this example, you will see how to implement the 6 basic steps to connect with database using JDBC in Java program.

Create Table
Before that, first, create one table and add some entries into it.

Below is the SQL query to create a table.
create table employee_details (empNum number(10), lastName varchar(50),

 firstName varchar(50), email varchar(255) , deptNum number(10), salary number(10));

Created the "employee_details" table in Oracle DB.



```
create table Employee_details (empNum number(10), lastName varchar2(50), firstName varchar2(50),
                                             email varchar(255), deptNum number(10),salary number(10));
```

```
Table EMPLOYEE_DETAILS created.
```

Insert Data Into Table
Using the following queries, insert the data into the "employee_details" table.

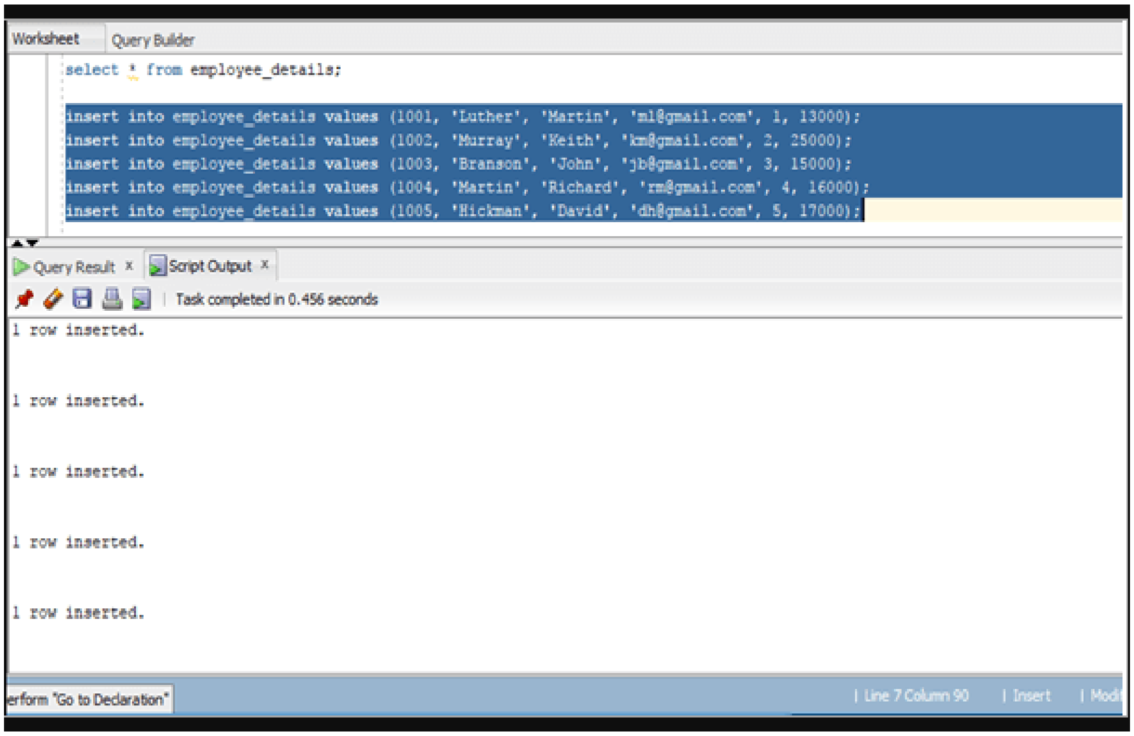
insert into employee_details values (1001, 'Luther', 'Martin', 'ml@gmail.com', 1, 13000);

insert into employee_details values (1002, 'Murray', 'Keith', 'km@gmail.com', 2, 25000);

insert into employee_details values (1003, 'Branson', 'John', 'jb@gmail.com', 3, 15000);

insert into employee_details values (1004, 'Martin', 'Richard', 'rm@gmail.com', 4, 16000);

insert into employee_details values (1005, 'Hickman', 'David', 'dh@gmail.com', 5, 17000);



```
select * from employee_details;

insert into employee_details values (1001, 'Luther', 'Martin', 'ml@gmail.com', 1, 13000);
insert into employee_details values (1002, 'Murray', 'Keith', 'km@gmail.com', 2, 25000);
insert into employee_details values (1003, 'Branson', 'John', 'jb@gmail.com', 3, 15000);
insert into employee_details values (1004, 'Martin', 'Richard', 'rm@gmail.com', 4, 16000);
insert into employee_details values (1005, 'Hickman', 'David', 'dh@gmail.com', 5, 17000);
```

Task completed in 0.456 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

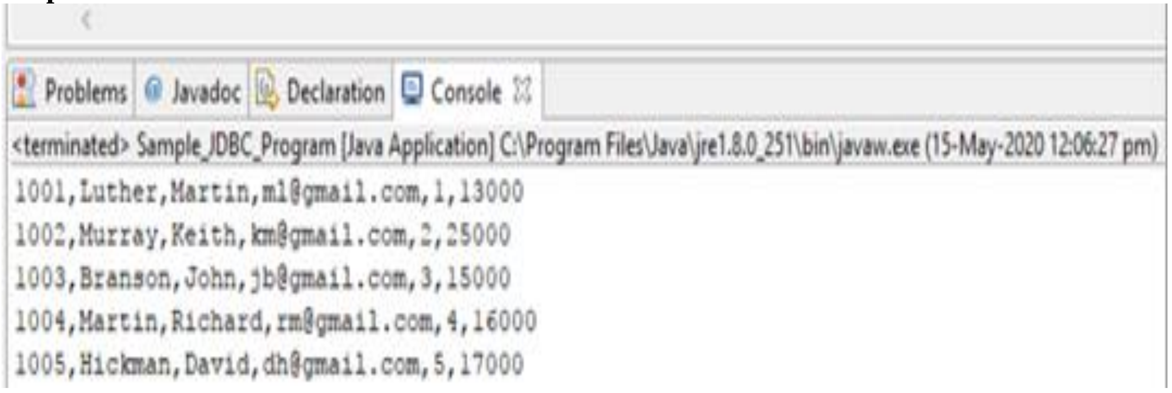
Java Program

Download the JDBC jar file and import it into the Java project.

```
package com.STH.JDBC;

  // import sql package to use it in our program

  import java.sql.*;


  public class Sample_JDBC_Program {


  public static void main(String[] args) throws ClassNotFoundException, SQLException {

    // store the SQL statement in a string

    String QUERY = "select * from employee_details";

    //register the oracle driver with DriverManager

    Class.forName("oracle.jdbc.driver.OracleDriver");
    //Here we have used Java 8 so opening the connection in try statement
  try(Connection conn = DriverManager.getConnection("jdbc:oracle:thin:system/pass123@localhost:1521:XE"))
    {
        Statement statemnt1 = conn.createStatement();
        //Created statement and execute it
        ResultSet rs1 = statemnt1.executeQuery(QUERY);
        {
          //Get the values of the record using while loop
          while(rs1.next())
          {
            int empNum = rs1.getInt("empNum");
            String lastName = rs1.getString("lastName");
            String firstName = rs1.getString("firstName");
            String email = rs1.getString("email");
            String deptNum = rs1.getString("deptNum");
            String salary = rs1.getString("salary");
            //store the values which are retrieved using ResultSet and print it
         System.out.println(empNum + "," +lastName+ "," +firstName+ "," +email +","+deptNum +"," +salary);
          }
        }
    }
    catch (SQLException e) {
      //If exception occurs catch it and exit the program
      e.printStackTrace();
```

```
        }
      }
    }
```

Output:



```
<terminated> Sample_JDBC_Program [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (15-May-2020 12:06:27 pm)
1001,Luther,Martin,ml@gmail.com,1,13000
1002,Murray,Keith,km@gmail.com,2,25000
1003,Branson,John,jb@gmail.com,3,15000
1004,Martin,Richard,rm@gmail.com,4,16000
1005,Hickman,David,dh@gmail.com,5,17000
```

Key points to be noted:

- First, we need to import the packages which we will be using in our Java program for the JDBC connection. So we can use the classes, subclasses, and interfaces in the packages.
- We need to register or load the driver with DriverManager before establishing a connection.
- After registering the driver, we can establish the connection and perform the operations.
- Using a statement interface we can create and execute the SQL query. For a simple SQL query, we can use the Statement interface. For insert/update/delete, we can use the PreparedStatement interface.
- After the statement execution, the results will be stored in the ResultSet object. We get the results from the ResultSet object using the next() method for more than 1 record.
- Once we are done with the database operation, we need to close the connection. So that the resource will be available for others to use.

What Are Database Applications?

“Database application” can mean two things:

One: It can refer to software running a database system. MongoDB Server or SQL Server are both software that provide the following:

1. Efficiently store and retrieve data from a file system to a network client.
2. Offer rich capabilities for querying and manipulating data from a variety of drivers.
3. Secure and authorize the access to the stored data
4. Scale
5. Provide fault tolerance and recovery (including backups) for our data

Two: It can refer to applications that are heavily coupled to a specific database and built to provide elements of that database to the end user. Some examples of such applications include:

- Online encyclopedias (Wikipedia)
- Social media websites (Facebook)
- CRM systems (Salesforce)
- Email systems (Gmail)
- E-commerce websites (Amazon)

The Purpose of Database Applications

The main purpose of database applications is to provide a way for data to be consumed either by end users (via UI) or other higher-level applications (via APIs). A database application can be used for storing or retrieving data, processing transactions, or various machine learning calculations.

For example, Facebook has a user database with which it authenticates users when they log into their Facebook account. However, Facebook also provides the ability to consume their user database by another application. This is done via a secure API Facebook exposes, and you could probably see this in many of today's platforms' authentication methods.

Another example is MongoDB Atlas, a Data-as-a-Service platform. Atlas clusters provide a variety of ways to consume the data — for example, via a driver, a Realm serverless function, or even via MongoDB Charts to provide dashboards based on data stored in Atlas.

Database Application Types (and their Pros and Cons)

Organizations and database administrators have to understand the pros and cons of the different database applications and database software out there. Databases can be categorized by the way they structure and consume data. Some use a normalized model and relations (Relational) while others use nested objects (Documents and some NoSQL flavors).

Database Application Type	Pros	Cons
Database Software - Document Store (eg., MongoDB)	• Flexible schema • Rich query language • Built-in resilience and scalability • Rich indexing strategies • Growing support communities and open-source projects • Transaction processing	• Learning curve for SQL-oriented developers • Relational schemas will need a redesign to work optimally
Database Software - Other NoSQL	• Distributed systems • More modern data stores	• Schemas are not flexible • Small support communities • Not general purpose - good for narrow use cases • No transaction processing
Database Software - Relational Databases (SQL)	• SQL-oriented • Large communities • Owned by large companies	• Expensive to start • Usually requires strong hardware to start • Not designed for the cloud era
Database Application Providers - (Amazon, Facebook)	• Offer robust services • Cloud-oriented	• Not flexible in the API • Limited ability to work with raw data • Not a pure database software

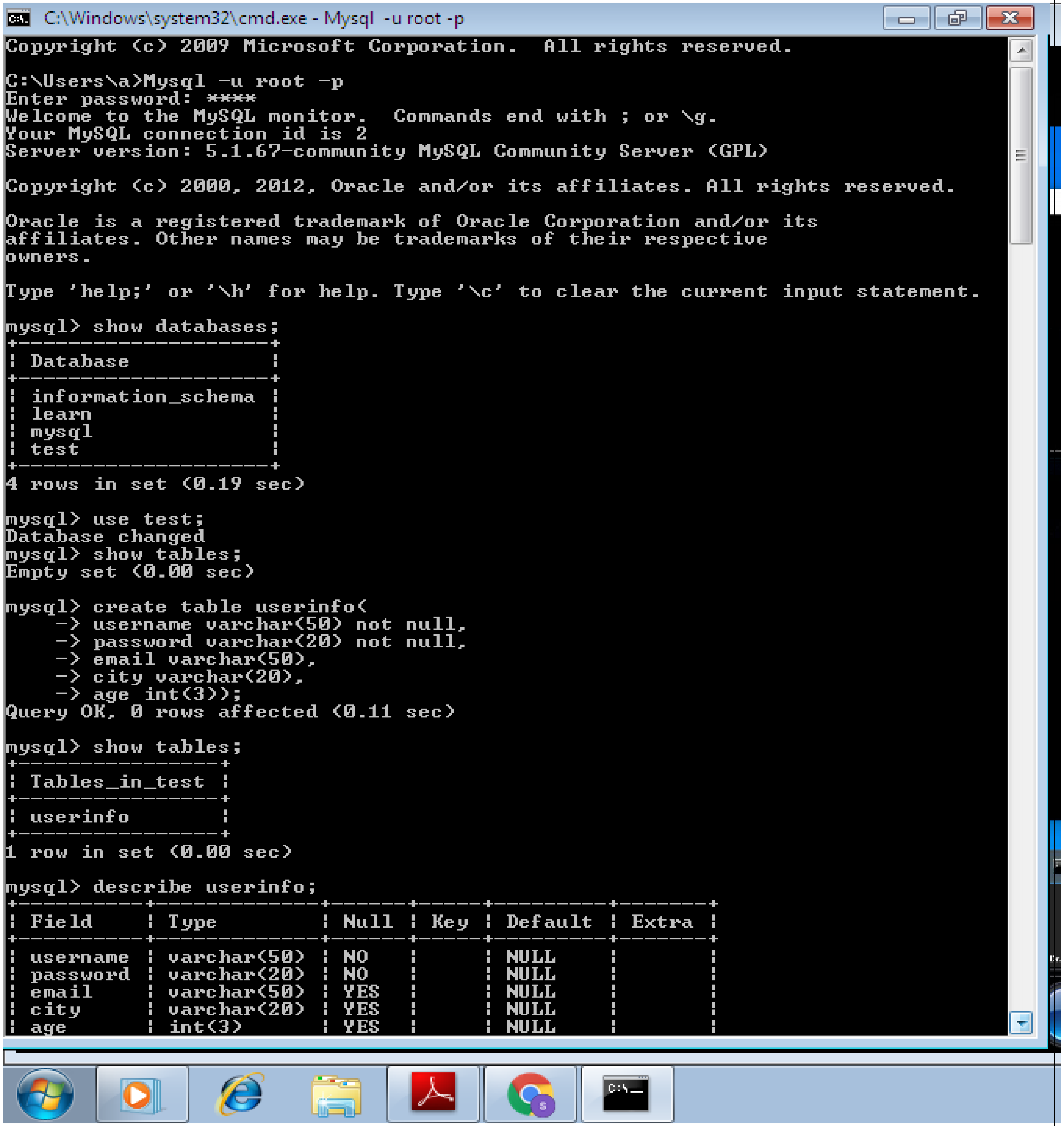
AJP Lab 8a. Insert the data from the database using JDBC

Program:

```
package Mysql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;


public class SQLPreparedStatementInsert {
public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","ro
ot");


PreparedStatement stmt = con.prepareStatement("insert into userinfo
values (?,?,?,?,?)");


stmt.setString(1,"user4");
stmt.setString(2,"pass4");
stmt.setString(3,"user4@gmail.com");
stmt.setString(4, "Nagpur");
stmt.setString(5, "60");

int i = stmt.executeUpdate();
System.out.println(i + "Records inserted..");
con.close();
}

catch(Exception e){
System.out.println(e);
}
}
}
```

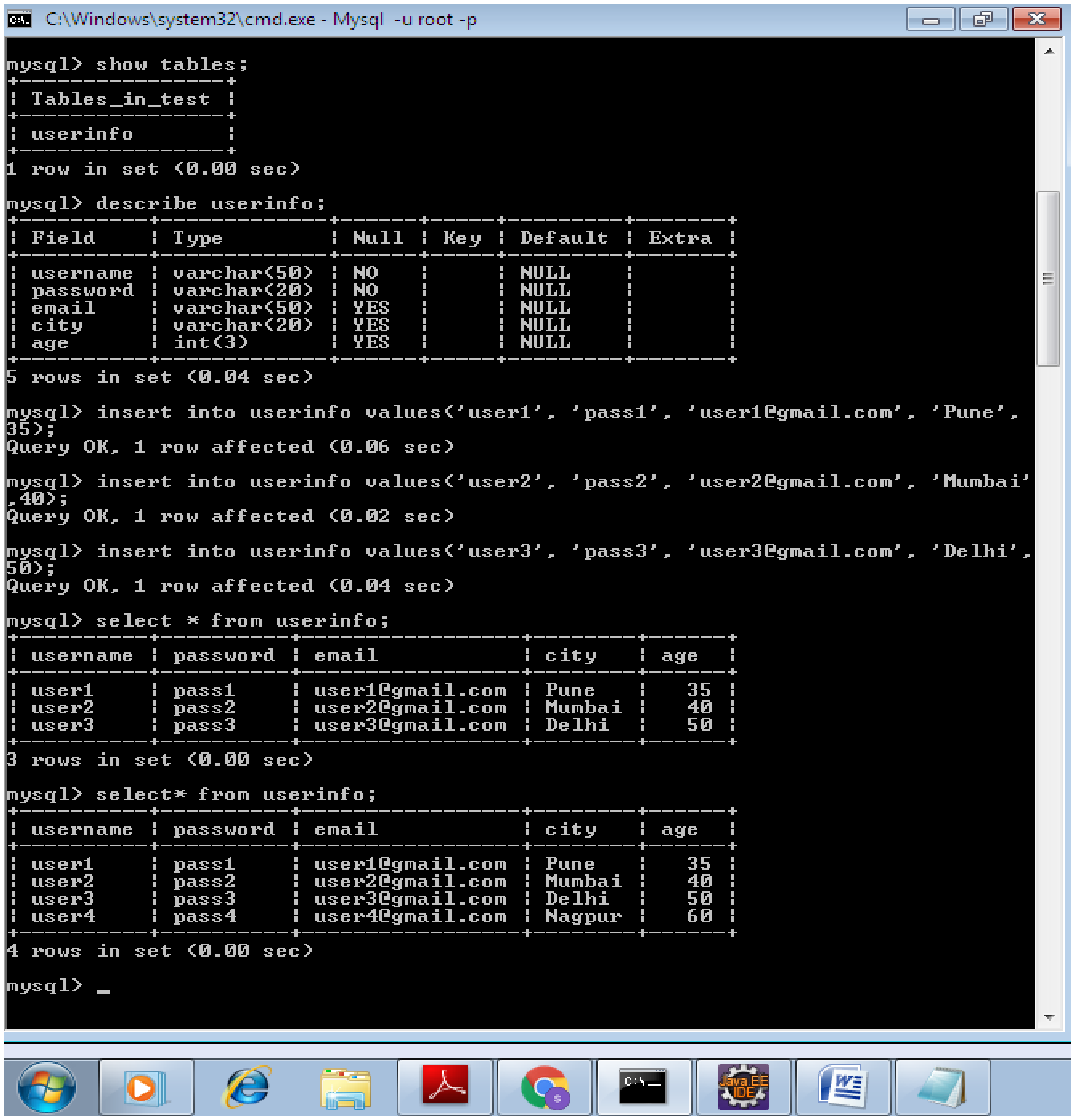
Output:

Before Insert program-

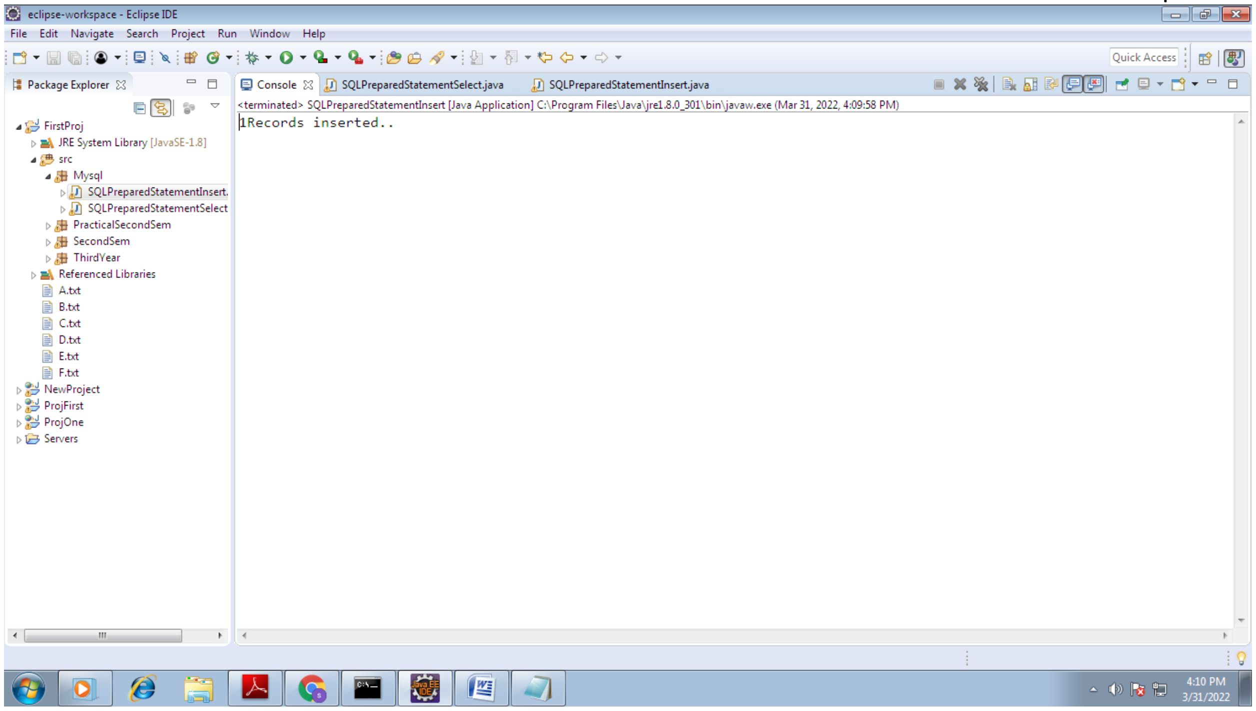

C:\Windows\system32\cmd.exe - Mysql -u root -p

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\a>Mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.67-community MySQL Community Server (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| learn              |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.19 sec)

mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table userinfo(
    -> username varchar(50) not null,
    -> password varchar(20) not null,
    -> email varchar(50),
    -> city varchar(20),
    -> age int(3));
Query OK, 0 rows affected (0.11 sec)

mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| userinfo       |
+----------------+
1 row in set (0.00 sec)

mysql> describe userinfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(50) | NO   |     | NULL    |       |
| password | varchar(20) | NO   |     | NULL    |       |
| email    | varchar(50) | YES  |     | NULL    |       |
| city     | varchar(20) | YES  |     | NULL    |       |
| age      | int(3)      | YES  |     | NULL    |       |
```

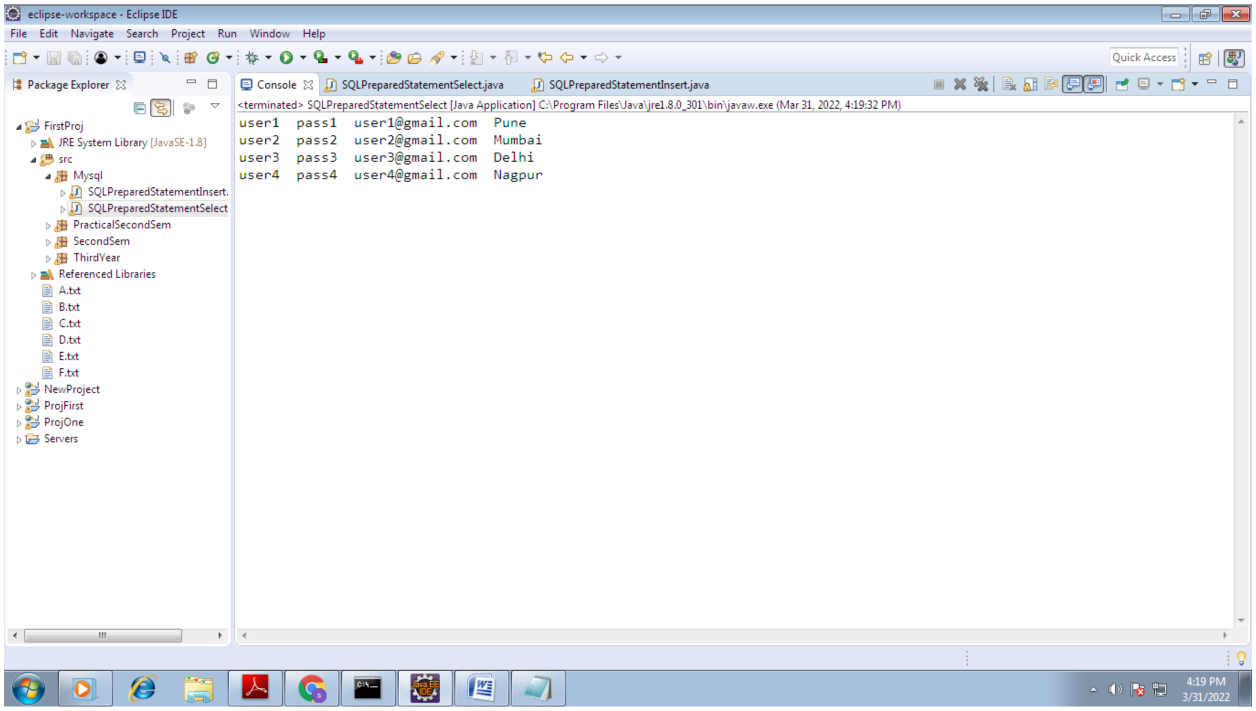
After Insert program-



```
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| userinfo       |
+----------------+
1 row in set (0.00 sec)

mysql> describe userinfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(50) | NO   |     | NULL    |       |
| password | varchar(20) | NO   |     | NULL    |       |
| email    | varchar(50) | YES  |     | NULL    |       |
| city     | varchar(20) | YES  |     | NULL    |       |
| age      | int(3)      | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
5 rows in set (0.04 sec)

mysql> insert into userinfo values('user1', 'pass1', 'user1@gmail.com', 'Pune',
35);
Query OK, 1 row affected (0.06 sec)

mysql> insert into userinfo values('user2', 'pass2', 'user2@gmail.com', 'Mumbai'
,40);
Query OK, 1 row affected (0.02 sec)

mysql> insert into userinfo values('user3', 'pass3', 'user3@gmail.com', 'Delhi',
50);
Query OK, 1 row affected (0.04 sec)

mysql> select * from userinfo;
+----------+----------+-----------------+--------+------+
| username | password | email           | city   | age  |
+----------+----------+-----------------+--------+------+
| user1    | pass1    | user1@gmail.com | Pune   |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai |   40 |
| user3    | pass3    | user3@gmail.com | Delhi  |   50 |
+----------+----------+-----------------+--------+------+
3 rows in set (0.00 sec)

mysql> select* from userinfo;
+----------+----------+-----------------+--------+------+
| username | password | email           | city   | age  |
+----------+----------+-----------------+--------+------+
| user1    | pass1    | user1@gmail.com | Pune   |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai |   40 |
| user3    | pass3    | user3@gmail.com | Delhi  |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur |   60 |
+----------+----------+-----------------+--------+------+
4 rows in set (0.00 sec)

mysql> _
```



```
<terminated> SQLPreparedStatementInsert [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Mar 31, 2022, 4:09:58 PM)
1Records inserted..
```



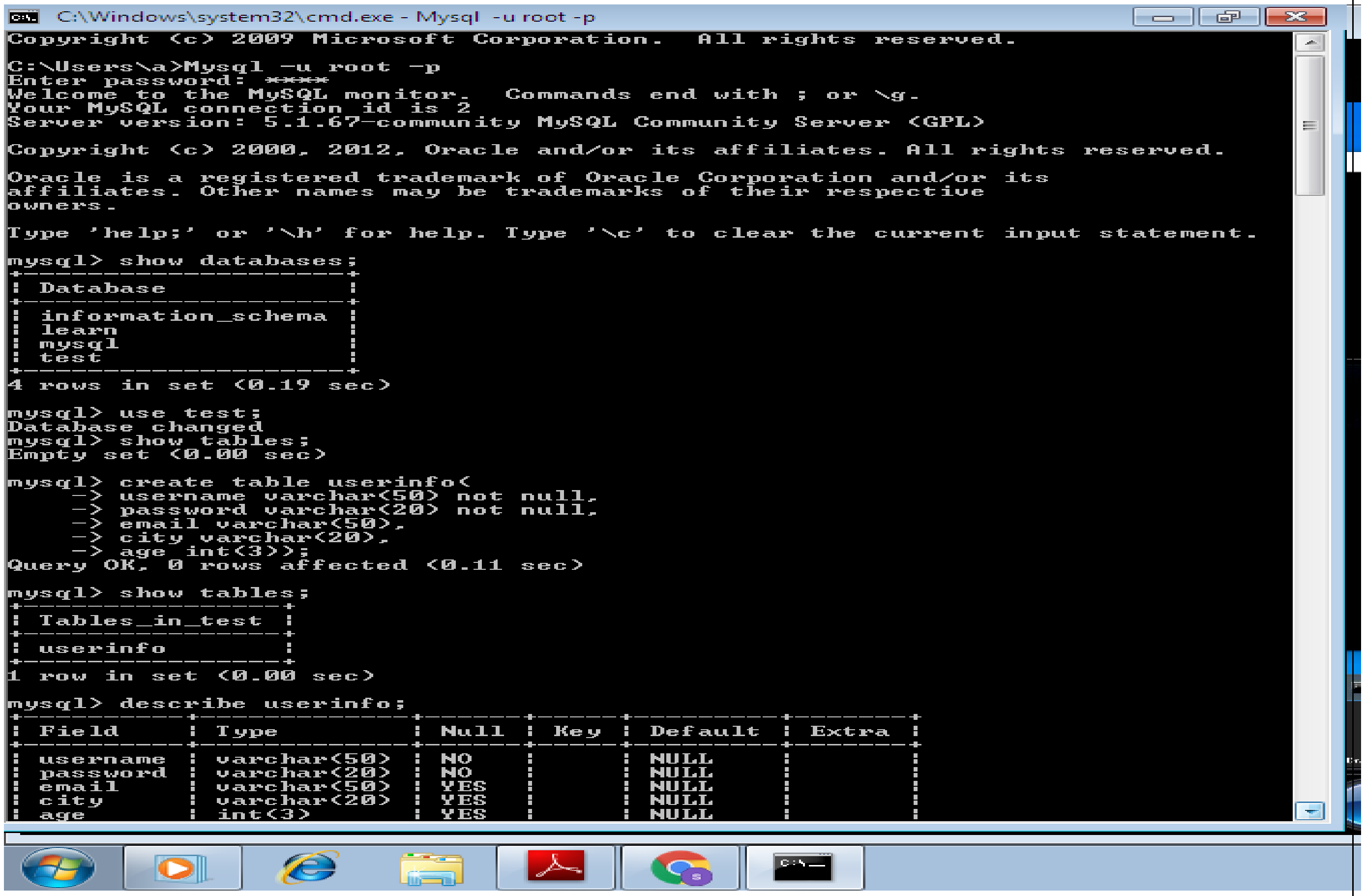
```
<terminated> SQLPreparedStatementSelect [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Mar 31, 2022, 4:19:32 PM)
user1  pass1  user1@gmail.com  Pune
user2  pass2  user2@gmail.com  Mumbai
user3  pass3  user3@gmail.com  Delhi
user4  pass4  user4@gmail.com  Nagpur
```

AJP Lab 8b. Rretrieve the data from the database using JDBC.

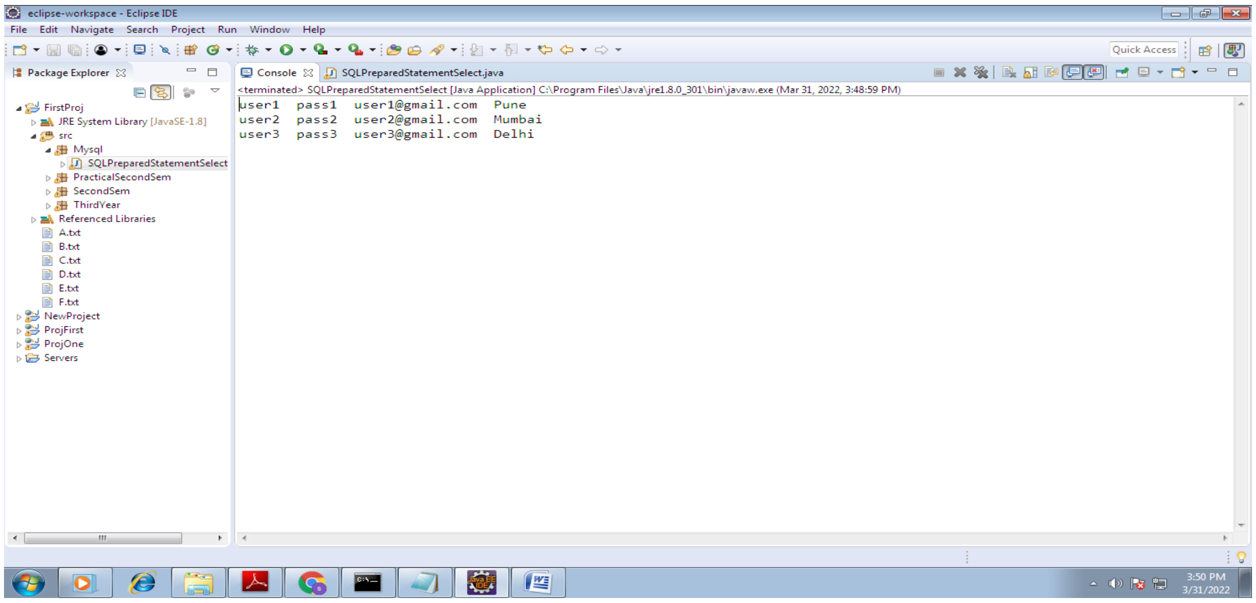
Program:

```
package Mysql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
public class SQLPreparedStatementSelect {

public static void main(String[] args) {
try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","ro
ot");


PreparedStatement stmt = con.prepareStatement("select * from userinfo");
ResultSet rs = stmt.executeQuery();
while(rs.next())
{

System.out.println(rs.getString(1) + "   " +
rs.getString(2) + "   " + rs.getString(3) + "   " +
rs.getString(4) + "   ");
}

con.close();
}

catch(Exception e){
System.out.println(e);
}
}
}
```

Output:



```
C:\Windows\system32\cmd.exe - Mysql -u root -p
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\a>Mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.67-community MySQL Community Server (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| learn              |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.19 sec)

mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table userinfo(
    -> username varchar(50) not null,
    -> password varchar(20) not null,
    -> email varchar(50),
    -> city varchar(20),
    -> age int(3));
Query OK, 0 rows affected (0.11 sec)

mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| userinfo       |
+----------------+
1 row in set (0.00 sec)

mysql> describe userinfo;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(50) | NO   |     | NULL    |       |
| password | varchar(20) | NO   |     | NULL    |       |
| email    | varchar(50) | YES  |     | NULL    |       |
| city     | varchar(20) | YES  |     | NULL    |       |
| age      | int(3)      | YES  |     | NULL    |       |
```



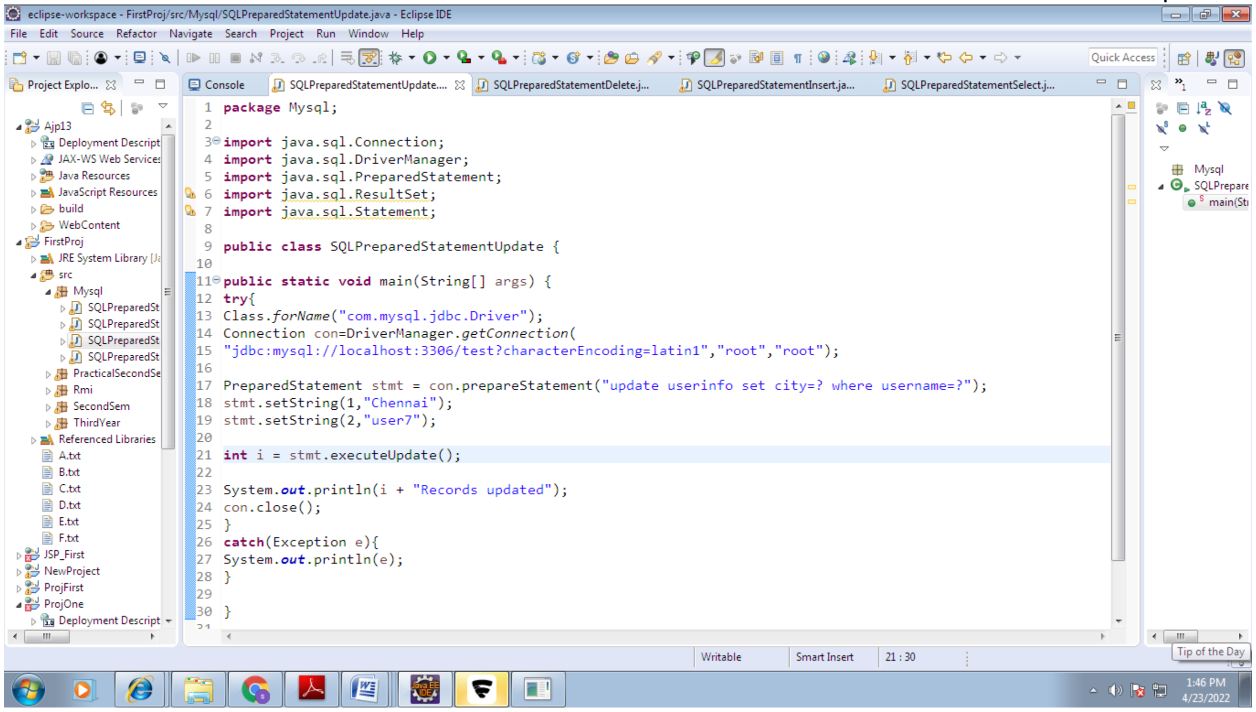
```
<terminated> SQLPreparedStatementSelect [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Mar 31, 2022, 3:48:59 PM)
user1  pass1  user1@gmail.com  Pune
user2  pass2  user2@gmail.com  Mumbai
user3  pass3  user3@gmail.com  Delhi
```

AJP Lab 8C. Update the data from the database using JDBC.

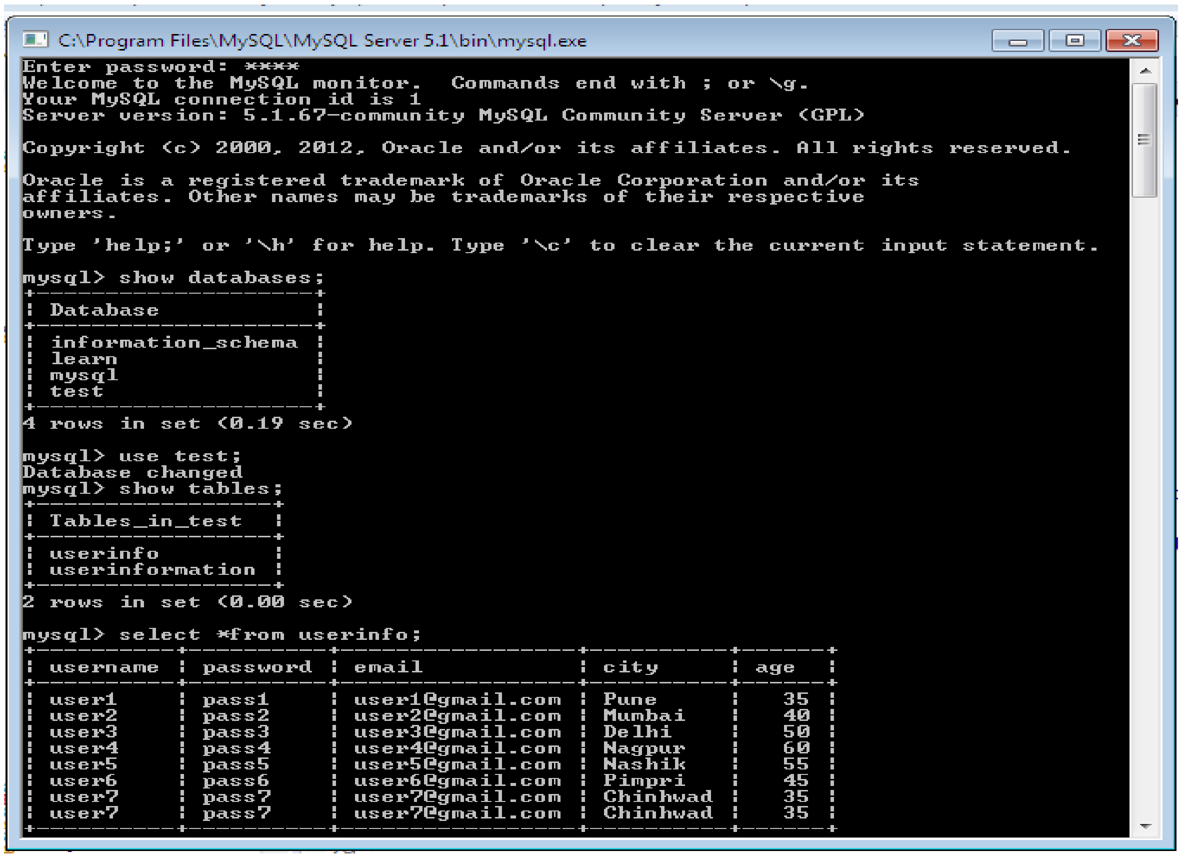
Program:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLPreparedStatementUpdate {

public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","root");

PreparedStatement stmt = con.prepareStatement("update userinfo set  city=? where username=?");
stmt.setString(1,"Chennai");
stmt.setString(2,"user7");
int i = stmt.executeUpdate();

System.out.println(i + "Records updated");
con.close();
}
catch(Exception e){
System.out.println(e);
}
}

}
```

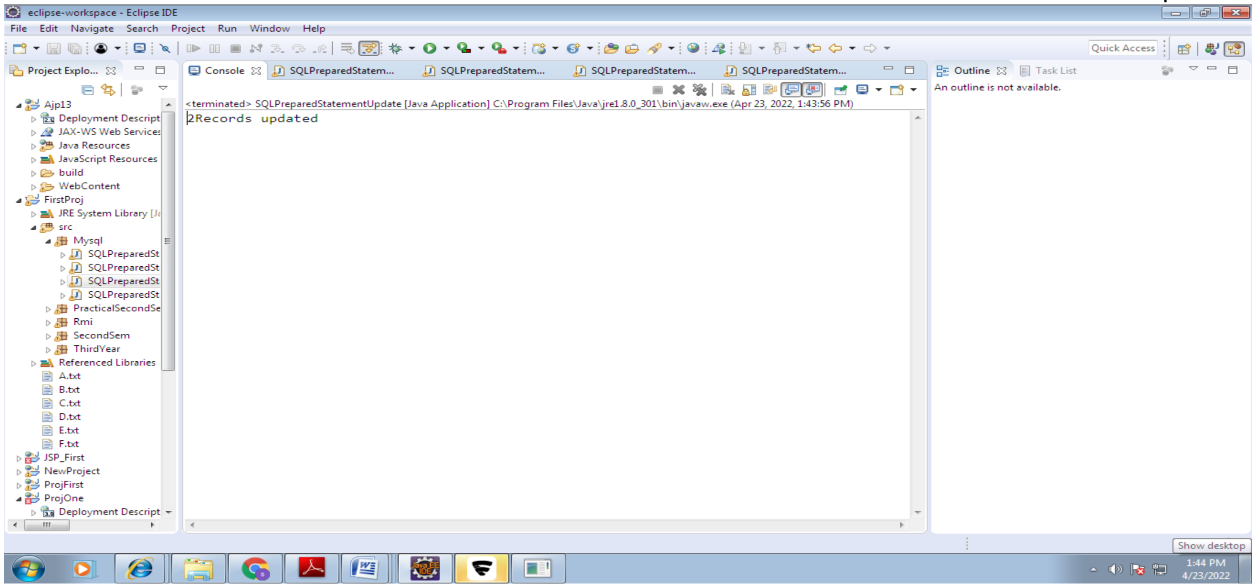
Output:


eclipse-workspace - FirstProj/src/Mysql/SQLPreparedStatementUpdate.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explo...
Console
SQLPreparedStatementUpdate....
SQLPreparedStatementDelete.j...
SQLPreparedStatementInsert.ja...
SQLPreparedStatementSelect.j...
Writable
Smart Insert
21 : 30

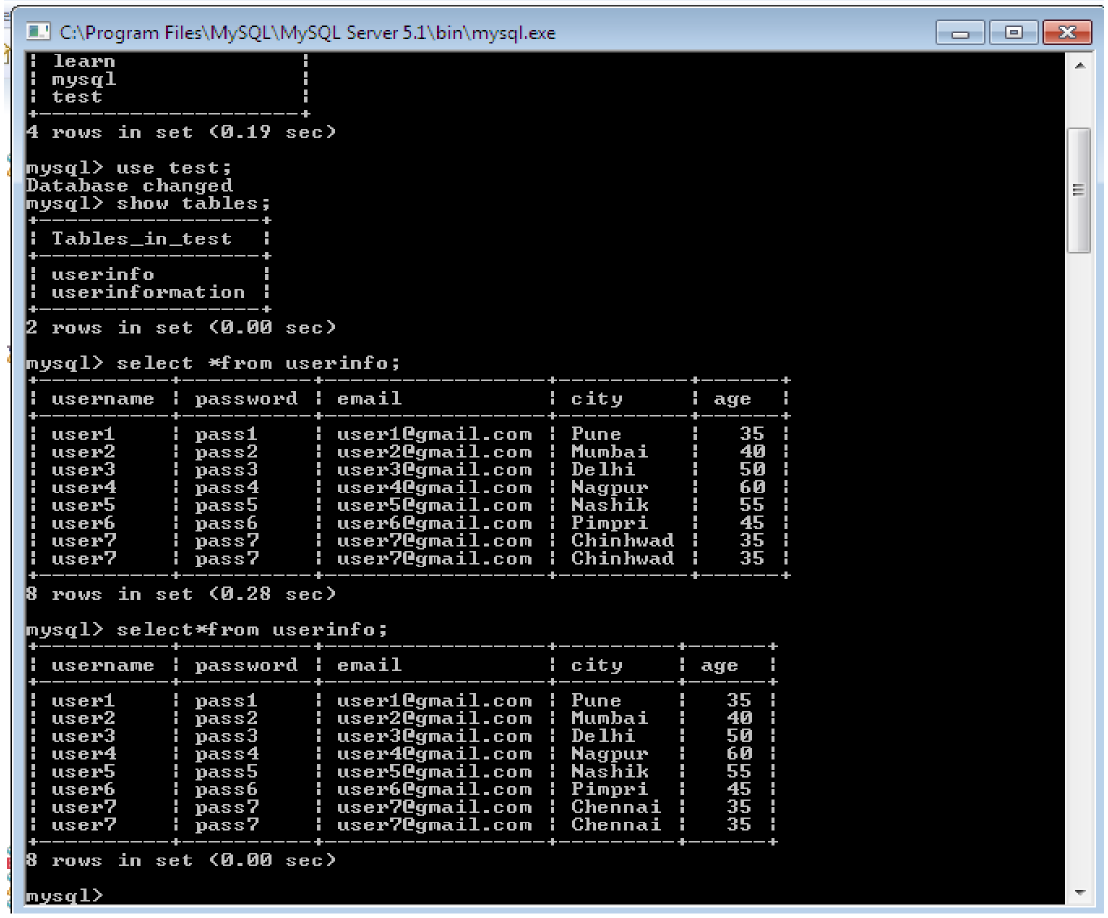
```java
package Mysql;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLPreparedStatementUpdate {

public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","root");

PreparedStatement stmt = con.prepareStatement("update userinfo set city=? where username=?");
stmt.setString(1,"Chennai");
stmt.setString(2,"user7");

int i = stmt.executeUpdate();

System.out.println(i + "Records updated");
con.close();
}
catch(Exception e){
System.out.println(e);
}

}
```



```
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.67-community MySQL Community Server (GPL)

Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.

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> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| learn              |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.19 sec)

mysql> use test;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| userinfo        |
| userinformation |
+-----------------+
2 rows in set (0.00 sec)

mysql> select *from userinfo;
+----------+----------+-----------------+----------+------+
| username | password | email           | city     | age  |
+----------+----------+-----------------+----------+------+
| user1    | pass1    | user1@gmail.com | Pune     |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai   |   40 |
| user3    | pass3    | user3@gmail.com | Delhi    |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur   |   60 |
| user5    | pass5    | user5@gmail.com | Nashik   |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri   |   45 |
| user7    | pass7    | user7@gmail.com | Chinhwad |   35 |
| user7    | pass7    | user7@gmail.com | Chinhwad |   35 |
+----------+----------+-----------------+----------+------+
```


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Project Explo...
Console
Outline
An outline is not available.

```
<terminated> SQLPreparedStatementUpdate [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Apr 23, 2022, 1:43:56 PM)
2Records updated
```

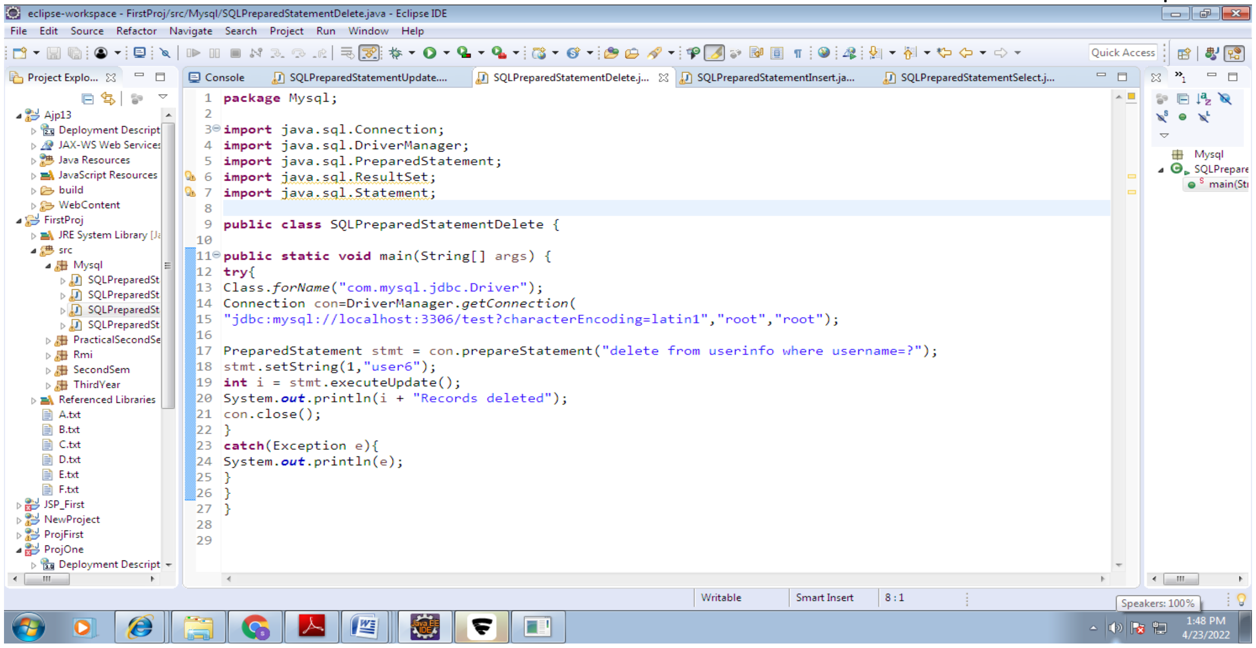


```
| learn              |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.19 sec)

mysql> use test;
Database changed
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| userinfo        |
| userinformation |
+-----------------+
2 rows in set (0.00 sec)

mysql> select *from userinfo;
+----------+----------+-----------------+----------+------+
| username | password | email           | city     | age  |
+----------+----------+-----------------+----------+------+
| user1    | pass1    | user1@gmail.com | Pune     |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai   |   40 |
| user3    | pass3    | user3@gmail.com | Delhi    |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur   |   60 |
| user5    | pass5    | user5@gmail.com | Nashik   |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri   |   45 |
| user7    | pass7    | user7@gmail.com | Chinhwad |   35 |
| user7    | pass7    | user7@gmail.com | Chinhwad |   35 |
+----------+----------+-----------------+----------+------+
8 rows in set (0.28 sec)

mysql> select*from userinfo;
+----------+----------+-----------------+---------+------+
| username | password | email           | city    | age  |
+----------+----------+-----------------+---------+------+
| user1    | pass1    | user1@gmail.com | Pune    |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai  |   40 |
| user3    | pass3    | user3@gmail.com | Delhi   |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur  |   60 |
| user5    | pass5    | user5@gmail.com | Nashik  |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri  |   45 |
| user7    | pass7    | user7@gmail.com | Chennai |   35 |
| user7    | pass7    | user7@gmail.com | Chennai |   35 |
+----------+----------+-----------------+---------+------+
8 rows in set (0.00 sec)

mysql>
```

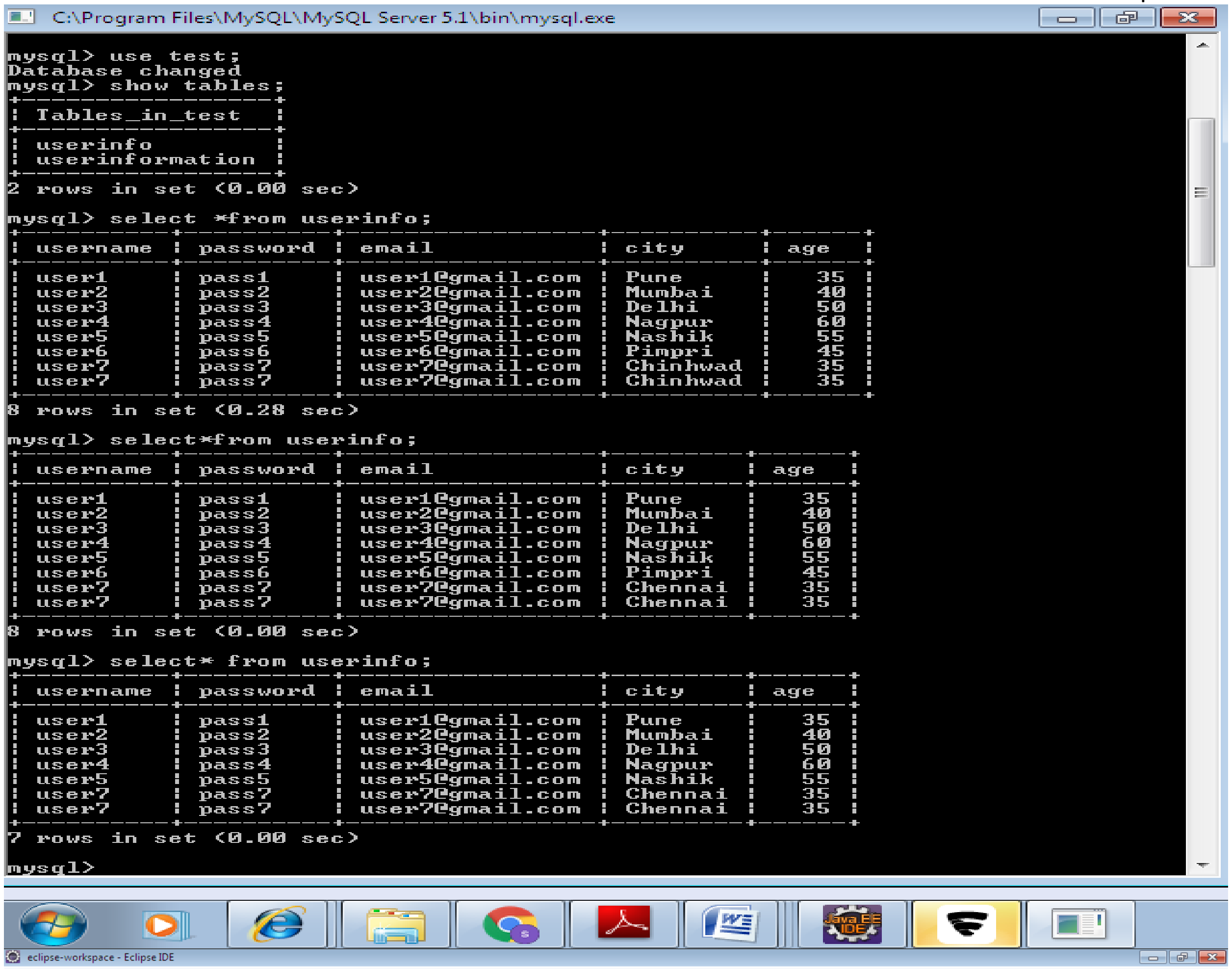
AJP Lab 8D. Delete the data from the database using JDBC.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLPreparedStatementDelete {

public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/logininfo?characterEncoding=latin1","root","root");

PreparedStatement stmt = con.prepareStatement("delete from userinfo where username=?");
stmt.setString(1,"user2");
int i = stmt.executeUpdate();
System.out.println(i + "Records deleted");
con.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```

Output:

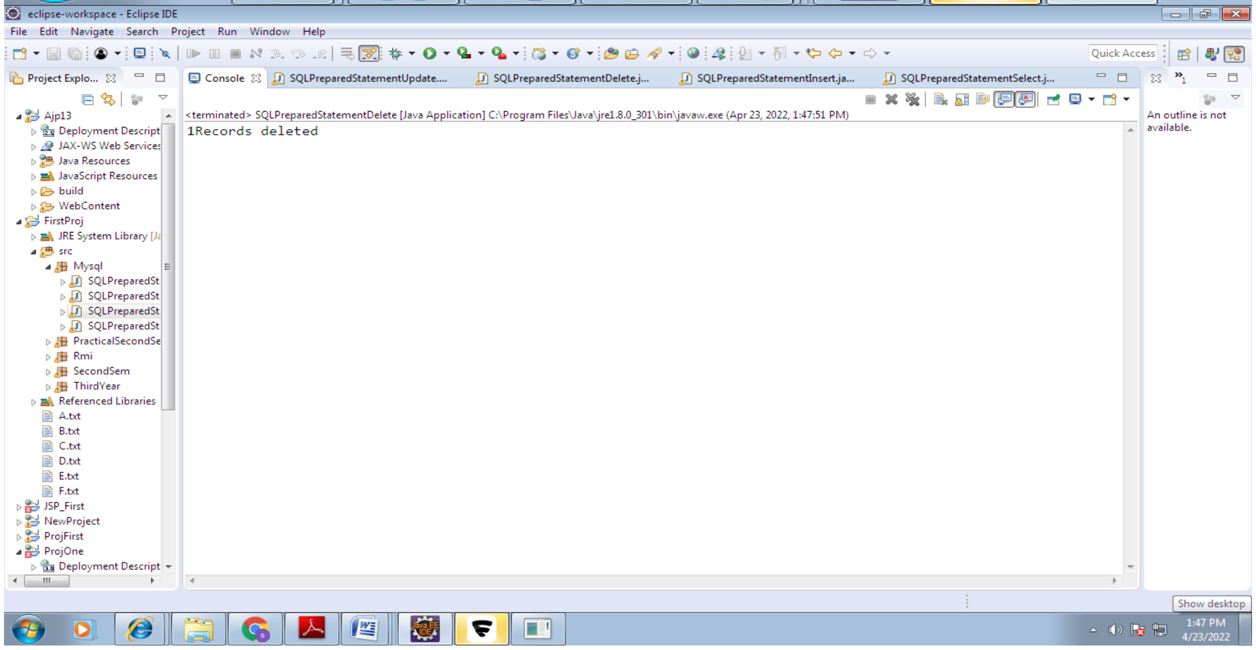

eclipse-workspace - FirstProj/src/Mysql/SQLPreparedStatementDelete.java - Eclipse IDE

```
package Mysql;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLPreparedStatementDelete {

public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","root");

PreparedStatement stmt = con.prepareStatement("delete from userinfo where username=?");
stmt.setString(1,"user6");
int i = stmt.executeUpdate();
System.out.println(i + "Records deleted");
con.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```



C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

```
mysql> use test;
Database changed
mysql> show tables;
+-------------------+
| Tables_in_test    |
+-------------------+
| userinfo          |
| userinformation   |
+-------------------+
2 rows in set (0.00 sec)

mysql> select *from userinfo;
+----------+----------+-----------------+-----------+------+
| username | password | email           | city      | age  |
+----------+----------+-----------------+-----------+------+
| user1    | pass1    | user1@gmail.com | Pune      |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai    |   40 |
| user3    | pass3    | user3@gmail.com | Delhi     |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur    |   60 |
| user5    | pass5    | user5@gmail.com | Nashik    |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri    |   45 |
| user7    | pass7    | user7@gmail.com | Chinhwad  |   35 |
| user7    | pass7    | user7@gmail.com | Chinhwad  |   35 |
+----------+----------+-----------------+-----------+------+
8 rows in set (0.28 sec)

mysql> select*from userinfo;
+----------+----------+-----------------+---------+------+
| username | password | email           | city    | age  |
+----------+----------+-----------------+---------+------+
| user1    | pass1    | user1@gmail.com | Pune    |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai  |   40 |
| user3    | pass3    | user3@gmail.com | Delhi   |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur  |   60 |
| user5    | pass5    | user5@gmail.com | Nashik  |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri  |   45 |
| user7    | pass7    | user7@gmail.com | Chennai |   35 |
| user7    | pass7    | user7@gmail.com | Chennai |   35 |
+----------+----------+-----------------+---------+------+
8 rows in set (0.00 sec)

mysql> select* from userinfo;
+----------+----------+-----------------+---------+------+
| username | password | email           | city    | age  |
+----------+----------+-----------------+---------+------+
| user1    | pass1    | user1@gmail.com | Pune    |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai  |   40 |
| user3    | pass3    | user3@gmail.com | Delhi   |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur  |   60 |
| user5    | pass5    | user5@gmail.com | Nashik  |   55 |
| user7    | pass7    | user7@gmail.com | Chennai |   35 |
| user7    | pass7    | user7@gmail.com | Chennai |   35 |
+----------+----------+-----------------+---------+------+
7 rows in set (0.00 sec)

mysql>
```



```
<terminated> SQLPreparedStatementDelete [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Apr 23, 2022, 1:47:51 PM)
1Records deleted
```

Conclusion:

References:

Herbert Schildt , “Java : The Complete Reference” Tata McGraw-Hill (7th Edition).

Questions:

1. What is JDBC driver?
2. What are the different types of JDBC drivers in Java? Explain each with an example.
3. Which JDBC driver is fastest and used more commonly?
4. What is DriverManager in JDBC?