**PR2:-**

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100),

email VARCHAR(100),

salary DOUBLE

);

import java.sql.\*;

public class EmployeeDatabase {

// Database URL, username, and password

static final String DB\_URL = "jdbc:mysql://localhost:3306/your\_database"; // Replace with your database URL

static final String USER = "your\_username"; // Replace with your MySQL username

static final String PASS = "your\_password"; // Replace with your MySQL password

public static void main(String[] args) {

// Connection and Statement objects

Connection conn = null;

Statement stmt = null;

try {

// Step 1: Establishing a connection to the database

conn = DriverManager.getConnection(DB\_URL, USER, PASS);

System.out.println("Connected to the database...");

// Step 2: Create a statement object

stmt = conn.createStatement();

// Step 3: Fetch employee records from the employee table

String selectQuery = "SELECT \* FROM employee";

ResultSet rs = stmt.executeQuery(selectQuery);

System.out.println("Employee Records:");

// Display fetched records

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String email = rs.getString("email");

double salary = rs.getDouble("salary");

System.out.println("ID: " + id + ", Name: " + name + ", Email: " + email + ", Salary: " + salary);

}

// Step 4: Insert a new employee record into the employee table

String insertQuery = "INSERT INTO employee (id, name, email, salary) VALUES (4, 'John Doe', 'john@example.com', 50000)";

int rowsAffected = stmt.executeUpdate(insertQuery);

System.out.println(rowsAffected + " record(s) inserted.");

} catch (SQLException e) {

// Handle SQL exception

e.printStackTrace();

} finally {

// Step 5: Close the resources

try {

if (stmt != null) stmt.close();

if (conn != null) conn.close();

} catch (SQLException se) {

se.printStackTrace();

}

}

}

}

**JDBC (Java Database Connectivity)** is an API provided by Java to connect and interact with databases. It allows Java applications to send SQL queries and updates to a relational database and retrieve results. JDBC provides a standard interface to interact with any database by abstracting the underlying database details.

**Key Concepts of JDBC:**

1. **JDBC Drivers**:
   * JDBC requires a database-specific driver that translates Java calls into database-specific calls. These drivers are provided by the database vendors, and they allow Java applications to connect to different databases like MySQL, Oracle, PostgreSQL, etc.
   * There are four types of JDBC drivers:
     + **Type 1 Driver (JDBC-ODBC Bridge)**: Uses ODBC (Open Database Connectivity) as the bridge to connect to databases. It’s now deprecated.
     + **Type 2 Driver (Native-API Driver)**: Converts JDBC calls directly into database-specific calls.
     + **Type 3 Driver (Network Protocol Driver)**: Uses a middleware server to translate JDBC calls to a specific database.
     + **Type 4 Driver (Thin Driver)**: A pure Java driver that directly translates JDBC calls into database-specific calls.