1. Write a Java program to connect to a MySQL database using JDBC.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class ConnectToDatabase {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        System.out.println("Attempting to connect to the database...");

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD)) {

            if (connection != null) {

                System.out.println("Connection successful! You are now connected to the MySQL database.");

            }

        } catch (SQLException e) {

            System.err.println("Connection failed!");

            System.err.println("Error: " + e.getMessage());

        }

    }

}

1. Create a Java class to insert student records into a database table.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class InsertStudent {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        String sql = "INSERT INTO students (id, name, percentage) VALUES (?, ?, ?)";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            // Set the values for the new student

            pstmt.setInt(1, 101);

            pstmt.setString(2, "Alice Smith");

            pstmt.setDouble(3, 88.5);

            int rowsAffected = pstmt.executeUpdate();

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

        } catch (SQLException e) {

            System.err.println("Error inserting student: " + e.getMessage());

        }

    }

}

1. Write a JDBC program to fetch and display all student records from the database.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class FetchAllStudents {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        String sql = "SELECT id, name, percentage FROM students";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             Statement stmt = connection.createStatement();

             ResultSet rs = stmt.executeQuery(sql)) {

            System.out.println("All Student Records:");

            System.out.println("------------------------------------");

            while (rs.next()) {

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

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

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

                System.out.printf("ID: %d, Name: %s, Percentage: %.2f\n", id, name, percentage);

            }

        } catch (SQLException e) {

            System.err.println("Error fetching students: " + e.getMessage());

        }

    }

}

1. Develop a program to search a student by ID using JDBC.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class SearchStudentById {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        int studentId = 101; // The ID to search for

        String sql = "SELECT id, name, percentage FROM students WHERE id = ?";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            pstmt.setInt(1, studentId);

            try (ResultSet rs = pstmt.executeQuery()) {

                if (rs.next()) {

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

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

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

                    System.out.println("Student found with ID " + id + ":");

                    System.out.printf("Name: %s, Percentage: %.2f\n", name, percentage);

                } else {

                    System.out.println("No student found with ID: " + studentId);

                }

            }

        } catch (SQLException e) {

            System.err.println("Error searching for student: " + e.getMessage());

        }

    }

}

1. Implement an update operation to modify student details in the database using JDBC.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class UpdateStudent {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        int studentId = 101; // The ID of the student to update

        double newPercentage = 95.0;

        String sql = "UPDATE students SET percentage = ? WHERE id = ?";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            pstmt.setDouble(1, newPercentage);

            pstmt.setInt(2, studentId);

            int rowsAffected = pstmt.executeUpdate();

            System.out.println(rowsAffected + " row(s) updated successfully.");

        } catch (SQLException e) {

            System.err.println("Error updating student: " + e.getMessage());

        }

    }

}

1. Write a Java program to delete a student record from the database using JDBC.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class DeleteStudent {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        int studentId = 101; // The ID of the student to delete

        String sql = "DELETE FROM students WHERE id = ?";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            pstmt.setInt(1, studentId);

            int rowsAffected = pstmt.executeUpdate();

            System.out.println(rowsAffected + " row(s) deleted successfully.");

        } catch (SQLException e) {

            System.err.println("Error deleting student: " + e.getMessage());

        }

    }

}

1. Design a Java application to perform all CRUD (Create, Read, Update, Delete) operations on an **Employee** table using JDBC.

program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;

/\*\*

 \* Q7. Design a Java application to perform all CRUD (Create, Read, Update, Delete)

 \* operations on an Employee table using JDBC.

 \*

 \* Before running, create the employees table in your MySQL database:

 \* CREATE TABLE employees (

 \* id INT PRIMARY KEY,

 \* name VARCHAR(255),

 \* department VARCHAR(255),

 \* salary DOUBLE

 \* );

 \*/

public class EmployeeCrudApp {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        int choice;

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD)) {

            System.out.println("Employee CRUD Application connected to database.");

            do {

                System.out.println("\nEmployee Management System Menu:");

                System.out.println("1. Add a new employee");

                System.out.println("2. View all employees");

                System.out.println("3. Update an employee's salary");

                System.out.println("4. Delete an employee");

                System.out.println("5. Exit");

                System.out.print("Enter your choice: ");

                try {

                    choice = scanner.nextInt();

                    scanner.nextLine(); // Consume newline

                    switch (choice) {

                        case 1: addEmployee(connection, scanner); break;

                        case 2: viewAllEmployees(connection); break;

                        case 3: updateEmployeeSalary(connection, scanner); break;

                        case 4: deleteEmployee(connection, scanner); break;

                        case 5: System.out.println("Exiting application. Goodbye!"); break;

                        default: System.out.println("Invalid choice. Please try again.");

                    }

                } catch (java.util.InputMismatchException e) {

                    System.out.println("Invalid input. Please enter a number.");

                    scanner.nextLine(); // Clear the invalid input

                    choice = 0;

                }

            } while (choice != 5);

        } catch (SQLException e) {

            System.err.println("Database connection or operation error: " + e.getMessage());

        } finally {

            scanner.close();

        }

    }

    private static void addEmployee(Connection conn, Scanner scanner) throws SQLException {

        System.out.print("Enter employee ID: ");

        int id = scanner.nextInt();

        scanner.nextLine(); // Consume newline

        System.out.print("Enter employee name: ");

        String name = scanner.nextLine();

        System.out.print("Enter department: ");

        String dept = scanner.nextLine();

        System.out.print("Enter salary: ");

        double salary = scanner.nextDouble();

        String sql = "INSERT INTO employees (id, name, department, salary) VALUES (?, ?, ?, ?)";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, id);

            pstmt.setString(2, name);

            pstmt.setString(3, dept);

            pstmt.setDouble(4, salary);

            int rowsAffected = pstmt.executeUpdate();

            System.out.println(rowsAffected + " employee(s) added.");

        }

    }

    private static void viewAllEmployees(Connection conn) throws SQLException {

        String sql = "SELECT \* FROM employees";

        try (Statement stmt = conn.createStatement();

             ResultSet rs = stmt.executeQuery(sql)) {

            System.out.println("ID\tName\t\tDepartment\tSalary");

            System.out.println("-------------------------------------------------");

            while (rs.next()) {

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

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

                String dept = rs.getString("department");

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

                System.out.printf("%d\t%s\t\t%s\t\t%.2f\n", id, name, dept, salary);

            }

        }

    }

    private static void updateEmployeeSalary(Connection conn, Scanner scanner) throws SQLException {

        System.out.print("Enter employee ID to update: ");

        int id = scanner.nextInt();

        System.out.print("Enter new salary: ");

        double newSalary = scanner.nextDouble();

        String sql = "UPDATE employees SET salary = ? WHERE id = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setDouble(1, newSalary);

            pstmt.setInt(2, id);

            int rowsAffected = pstmt.executeUpdate();

            System.out.println(rowsAffected + " employee(s) updated.");

        }

    }

    private static void deleteEmployee(Connection conn, Scanner scanner) throws SQLException {

        System.out.print("Enter employee ID to delete: ");

        int id = scanner.nextInt();

        String sql = "DELETE FROM employees WHERE id = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {

            pstmt.setInt(1, id);

            int rowsAffected = pstmt.executeUpdate();

            System.out.println(rowsAffected + " employee(s) deleted.");

        }

    }

}

1. Create a JDBC-based program to count the total number of rows in a table.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.util.Scanner;

/\*\*

 \* Q8. Create a JDBC-based program to count the total number of rows in a table.

 \*/

public class CountRows {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the table name to count rows: ");

        String tableName = scanner.nextLine();

        scanner.close();

        String sql = "SELECT COUNT(\*) AS rowCount FROM " + tableName;

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             Statement stmt = connection.createStatement();

             ResultSet rs = stmt.executeQuery(sql)) {

            if (rs.next()) {

                int count = rs.getInt("rowCount");

                System.out.println("Total number of rows in '" + tableName + "': " + count);

            }

        } catch (SQLException e) {

            System.err.println("Error counting rows. Check if the table name is correct: " + e.getMessage());

        }

    }

}

1. Develop a program to sort student data in ascending order by name using SQL in JDBC.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

/\*\*

 \* Program to sort student data in ascending order by name using SQL in JDBC.

 \*

 \* Before running this, ensure you have a MySQL database running with a

 \* 'students' table and some data in it.

 \*

 \* SQL to create the table:

 \* CREATE TABLE students (

 \* id INT PRIMARY KEY,

 \* name VARCHAR(255),

 \* percentage DOUBLE

 \* );

 \*

 \* SQL to insert some sample data:

 \* INSERT INTO students VALUES (101, 'Charlie Brown', 72.5);

 \* INSERT INTO students VALUES (102, 'Alice Smith', 88.5);

 \* INSERT INTO students VALUES (103, 'David Lee', 81.3);

 \*/

public class SortStudentsByName {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        // The SQL query with the ORDER BY clause

        String sql = "SELECT id, name, percentage FROM students ORDER BY name ASC";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             Statement stmt = connection.createStatement();

             ResultSet rs = stmt.executeQuery(sql)) {

            System.out.println("Students sorted by name (A-Z):");

            System.out.println("------------------------------------");

            while (rs.next()) {

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

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

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

                System.out.printf("ID: %d, Name: %s, Percentage: %.2f\n", id, name, percentage);

            }

        } catch (SQLException e) {

            System.err.println("Error sorting students: " + e.getMessage());

        }

    }

}

1. Write a program to display all students whose percentage is greater than 75 using JDBC and SQL WHERE clause.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class FilterStudentsByPercentage {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        // The minimum percentage to filter by

        double minPercentage = 75.0;

        // The SQL query with a parameterized WHERE clause

        String sql = "SELECT id, name, percentage FROM students WHERE percentage > ?";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            // Set the parameter value for the prepared statement

            pstmt.setDouble(1, minPercentage);

            try (ResultSet rs = pstmt.executeQuery()) {

                System.out.println("Students with percentage greater than " + minPercentage + ":");

                System.out.println("------------------------------------");

                while (rs.next()) {

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

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

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

                    System.out.printf("ID: %d, Name: %s, Percentage: %.2f\n", id, name, percentage);

                }

            }

        } catch (SQLException e) {

            System.err.println("Error filtering students: " + e.getMessage());

        }

    }

}

1. Use **PreparedStatement** to insert multiple student records into the database.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class BatchInsertStudents {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        String sql = "INSERT INTO students (id, name, percentage) VALUES (?, ?, ?)";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            // Add the first student to the batch

            pstmt.setInt(1, 105);

            pstmt.setString(2, "Emily Davis");

            pstmt.setDouble(3, 91.2);

            pstmt.addBatch();

            // Add the second student to the batch

            pstmt.setInt(1, 106);

            pstmt.setString(2, "Frank Green");

            pstmt.setDouble(3, 78.9);

            pstmt.addBatch();

            // Add the third student to the batch

            pstmt.setInt(1, 107);

            pstmt.setString(2, "Grace Miller");

            pstmt.setDouble(3, 85.5);

            pstmt.addBatch();

            System.out.println("Executing batch insert...");

            int[] rowsAffected = pstmt.executeBatch();

            System.out.println("Batch insert successful. " + rowsAffected.length + " students inserted.");

        } catch (SQLException e) {

            System.err.println("Error during batch insert: " + e.getMessage());

        }

    }

}

1. Implement a program using **transaction management** in JDBC (i.e., commit and rollback).

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class TransactionExample {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD)) {

            // 1. Disable auto-commit to begin the transaction

            connection.setAutoCommit(false);

            try {

                // Transfer $200 from account 1 to account 2

                transferMoney(connection, 1, 2, 200.00);

                // 2. If no errors, commit the transaction

                connection.commit();

                System.out.println("Transaction committed successfully.");

            } catch (SQLException e) {

                // 3. If an error occurs, rollback the transaction

                System.err.println("Transaction failed, rolling back changes.");

                connection.rollback();

                System.err.println("Error: " + e.getMessage());

            } finally {

                // 4. Re-enable auto-commit

                connection.setAutoCommit(true);

            }

        } catch (SQLException e) {

            System.err.println("Database connection error: " + e.getMessage());

        }

    }

    private static void transferMoney(Connection conn, int fromId, int toId, double amount) throws SQLException {

        // Debit the 'from' account

        String debitSql = "UPDATE accounts SET balance = balance - ? WHERE id = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(debitSql)) {

            pstmt.setDouble(1, amount);

            pstmt.setInt(2, fromId);

            pstmt.executeUpdate();

        }

        // Credit the 'to' account

        String creditSql = "UPDATE accounts SET balance = balance + ? WHERE id = ?";

        try (PreparedStatement pstmt = conn.prepareStatement(creditSql)) {

            pstmt.setDouble(1, amount);

            pstmt.setInt(2, toId);

            pstmt.executeUpdate();

        }

        System.out.println("Money transfer operation complete.");

    }

}

1. Write a JDBC program to handle exceptions (like invalid ID, connection errors) gracefully.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class ExceptionHandling {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        // Example 1: Simulate an invalid database URL to cause a connection error

        try {

            System.out.println("Attempting to connect with an invalid URL...");

            Connection conn = DriverManager.getConnection("jdbc:invalid\_url", USER, PASSWORD);

            conn.close();

        } catch (SQLException e) {

            System.err.println("Error connecting to database (Example 1).");

            System.err.println("SQL State: " + e.getSQLState());

            System.err.println("Error Code: " + e.getErrorCode());

            System.err.println("Message: " + e.getMessage());

        }

        System.out.println("\n------------------------------------\n");

        // Example 2: Simulate an operation with a non-existent column

        String sql = "UPDATE students SET invalid\_column = ? WHERE id = ?";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            System.out.println("Attempting to execute an invalid SQL query...");

            pstmt.setDouble(1, 99.9);

            pstmt.setInt(2, 101);

            pstmt.executeUpdate();

        } catch (SQLException e) {

            System.err.println("Error updating student (Example 2).");

            System.err.println("SQL State: " + e.getSQLState());

            System.err.println("Error Code: " + e.getErrorCode());

            System.err.println("Message: " + e.getMessage());

        }

    }

}

1. Create a login system using JDBC where user credentials are verified from the database.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Scanner;

public class LoginSystem {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.println("--- User Login System ---");

        System.out.print("Enter username: ");

        String username = scanner.nextLine();

        System.out.print("Enter password: ");

        String password = scanner.nextLine();

        // The SQL query with placeholders for the username and password

        String sql = "SELECT username FROM users WHERE username = ? AND password = ?";

        boolean loginSuccessful = false;

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql)) {

            // Set the values for the placeholders

            pstmt.setString(1, username);

            pstmt.setString(2, password);

            // Execute the query

            try (ResultSet rs = pstmt.executeQuery()) {

                // If a row is returned, the credentials are valid

                if (rs.next()) {

                    loginSuccessful = true;

                }

            }

        } catch (SQLException e) {

            System.err.println("Database error: " + e.getMessage());

        } finally {

            scanner.close();

        }

        // Display the login result

        if (loginSuccessful) {

            System.out.println("\nLogin successful! Welcome, " + username + ".");

        } else {

            System.out.println("\nLogin failed. Invalid username or password.");

        }

    }

}

1. Implement a Java application to take dynamic input from the user and perform insertion, search, or update using menu-driven logic.

Program:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.Scanner;

public class StudentManagementApp {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    public static void main(String[] args) {

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             Scanner scanner = new Scanner(System.in)) {

            int choice;

            do {

                displayMenu();

                System.out.print("Enter your choice: ");

                choice = scanner.nextInt();

                scanner.nextLine(); // Consume newline left-over

                switch (choice) {

                    case 1:

                        insertStudent(connection, scanner);

                        break;

                    case 2:

                        searchStudent(connection, scanner);

                        break;

                    case 3:

                        updateStudent(connection, scanner);

                        break;

                    case 4:

                        System.out.println("Exiting application. Goodbye!");

                        break;

                    default:

                        System.out.println("Invalid choice. Please try again.");

                        break;

                }

                System.out.println(); // Add a blank line for readability

            } while (choice != 4);

        } catch (SQLException e) {

            System.err.println("Database error: " + e.getMessage());

        }

    }

    private static void displayMenu() {

        System.out.println("--- Student Management System ---");

        System.out.println("1. Insert a new student");

        System.out.println("2. Search for a student by ID");

        System.out.println("3. Update a student's percentage");

        System.out.println("4. Exit");

        System.out.println("---------------------------------");

    }

    private static void insertStudent(Connection connection, Scanner scanner) throws SQLException {

        System.out.print("Enter student ID: ");

        int id = scanner.nextInt();

        scanner.nextLine(); // Consume newline

        System.out.print("Enter student name: ");

        String name = scanner.nextLine();

        System.out.print("Enter student percentage: ");

        double percentage = scanner.nextDouble();

        String sql = "INSERT INTO students (id, name, percentage) VALUES (?, ?, ?)";

        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

            pstmt.setInt(1, id);

            pstmt.setString(2, name);

            pstmt.setDouble(3, percentage);

            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {

                System.out.println("Student inserted successfully!");

            } else {

                System.out.println("Insertion failed.");

            }

        }

    }

    private static void searchStudent(Connection connection, Scanner scanner) throws SQLException {

        System.out.print("Enter student ID to search: ");

        int id = scanner.nextInt();

        String sql = "SELECT id, name, percentage FROM students WHERE id = ?";

        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

            pstmt.setInt(1, id);

            try (ResultSet rs = pstmt.executeQuery()) {

                if (rs.next()) {

                    System.out.println("\nStudent found:");

                    System.out.printf("ID: %d, Name: %s, Percentage: %.2f\n",

                            rs.getInt("id"), rs.getString("name"), rs.getDouble("percentage"));

                } else {

                    System.out.println("Student with ID " + id + " not found.");

                }

            }

        }

    }

    private static void updateStudent(Connection connection, Scanner scanner) throws SQLException {

        System.out.print("Enter student ID to update: ");

        int id = scanner.nextInt();

        System.out.print("Enter new percentage: ");

        double percentage = scanner.nextDouble();

        String sql = "UPDATE students SET percentage = ? WHERE id = ?";

        try (PreparedStatement pstmt = connection.prepareStatement(sql)) {

            pstmt.setDouble(1, percentage);

            pstmt.setInt(2, id);

            int rowsAffected = pstmt.executeUpdate();

            if (rowsAffected > 0) {

                System.out.println("Student updated successfully!");

            } else {

                System.out.println("Update failed. Student with ID " + id + " not found.");

            }

        }

    }

}

1. Design the schema for a **Library Management System** and write JDBC programs for:

* Adding a book
* Viewing all books
* Issuing a book to a member
* Returning a book

Program:

-- Schema for a simple Library Management System

-- Create the Books table

-- This table stores information about each book in the library.

CREATE TABLE Books (

    book\_id INT PRIMARY KEY AUTO\_INCREMENT,

    title VARCHAR(255) NOT NULL,

    author VARCHAR(255) NOT NULL,

    year\_published INT,

    is\_available BOOLEAN DEFAULT TRUE

);

-- Create the Members table

-- This table stores the details of each library member.

CREATE TABLE Members (

    member\_id INT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(255) NOT NULL,

    email VARCHAR(255) UNIQUE NOT NULL

);

-- Create the Transactions table

-- This table tracks the issuance and return of books.

-- The ON DELETE CASCADE ensures that if a member or book is deleted,

-- their corresponding transaction records are also removed.

CREATE TABLE Transactions (

    transaction\_id INT PRIMARY KEY AUTO\_INCREMENT,

    book\_id INT,

    member\_id INT,

    issue\_date DATE NOT NULL,

    return\_date DATE,

    FOREIGN KEY (book\_id) REFERENCES Books(book\_id) ON DELETE CASCADE,

    FOREIGN KEY (member\_id) REFERENCES Members(member\_id) ON DELETE CASCADE

);

1. Create a **Hospital Management System** database. Using JDBC, implement:

* Register new patient
* Assign doctor
* Generate billing

Program:

-- Schema for a simple Hospital Management System with Billing

-- Create the Doctors table

-- This table stores information about the doctors.

CREATE TABLE Doctors (

    doctor\_id INT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(255) NOT NULL,

    specialization VARCHAR(100),

    fee DECIMAL(10, 2)

);

-- Create the Patients table

-- This table stores information about the patients.

CREATE TABLE Patients (

    patient\_id INT PRIMARY KEY AUTO\_INCREMENT,

    name VARCHAR(255) NOT NULL,

    dob DATE,

    contact\_number VARCHAR(20)

);

-- Create the Patient\_Doctor\_Assignments table

-- This table tracks which doctor is assigned to a patient.

CREATE TABLE Patient\_Doctor\_Assignments (

    assignment\_id INT PRIMARY KEY AUTO\_INCREMENT,

    patient\_id INT,

    doctor\_id INT,

    assignment\_date DATE NOT NULL,

    FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id) ON DELETE CASCADE,

    FOREIGN KEY (doctor\_id) REFERENCES Doctors(doctor\_id) ON DELETE CASCADE

);

-- Create a Billing table to manage patient charges

CREATE TABLE Billing (

    bill\_id INT PRIMARY KEY AUTO\_INCREMENT,

    patient\_id INT,

    service\_description VARCHAR(255) NOT NULL,

    amount DECIMAL(10, 2) NOT NULL,

    bill\_date DATE NOT NULL,

    FOREIGN KEY (patient\_id) REFERENCES Patients(patient\_id) ON DELETE CASCADE

);

1. Write a JDBC-based report generator that exports data from a MySQL table to a text or CSV file.

Program:  
import java.io.FileWriter;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

public class ReportGenerator {

    // Database connection details

    private static final String JDBC\_URL = "jdbc:mysql://localhost:3306/mydatabase";

    private static final String USER = "root"; // Change to your MySQL username

    private static final String PASSWORD = "password"; // Change to your MySQL password

    // File name for the report

    private static final String REPORT\_FILE = "student\_report.csv";

    public static void main(String[] args) {

        // SQL query to retrieve data from the 'students' table

        String sql = "SELECT id, name, percentage FROM students";

        try (Connection connection = DriverManager.getConnection(JDBC\_URL, USER, PASSWORD);

             PreparedStatement pstmt = connection.prepareStatement(sql);

             ResultSet rs = pstmt.executeQuery()) {

            // Create a FileWriter to write to the CSV file

            FileWriter writer = new FileWriter(REPORT\_FILE);

            // Get metadata to retrieve column names

            ResultSetMetaData rsmd = rs.getMetaData();

            int columnCount = rsmd.getColumnCount();

            // Write the header row (column names) to the CSV file

            for (int i = 1; i <= columnCount; i++) {

                writer.append(rsmd.getColumnName(i));

                if (i < columnCount) {

                    writer.append(",");

                }

            }

            writer.append("\n");

            // Write data rows to the CSV file

            while (rs.next()) {

                for (int i = 1; i <= columnCount; i++) {

                    // Get the data from the current column

                    String columnValue = rs.getString(i);

                    // Handle potential null values by writing an empty string

                    if (columnValue == null) {

                        writer.append("");

                    } else {

                        // Enclose values with commas in double quotes

                        if (columnValue.contains(",")) {

                            writer.append("\"" + columnValue + "\"");

                        } else {

                            writer.append(columnValue);

                        }

                    }

                    if (i < columnCount) {

                        writer.append(",");

                    }

                }

                writer.append("\n");

            }

            writer.flush();

            writer.close();

            System.out.println("Report successfully generated to " + REPORT\_FILE);

        } catch (SQLException e) {

            System.err.println("Database error: " + e.getMessage());

        } catch (IOException e) {

            System.err.println("File writing error: " + e.getMessage());

        }

    }

}