**Day-11 Assignment**

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

ANS:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class MySQLConnectionExample {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/yourDatabase";

String user = "root";

String password = "yourPassword";

try {

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

System.out.println("Connection successful!");

conn.close();

} catch (SQLException e) {

System.out.println("Connection failed: " + e.getMessage());

}

}

}

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

ANS:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class StudentInsert {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/yourDatabase";

String user = "root";

String password = "yourPassword";

String insertSQL = "INSERT INTO students (id, name, marks) VALUES (?, ?, ?)";

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

PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

pstmt.setInt(1, 101);

pstmt.setString(2, "Alice");

pstmt.setInt(3, 85);

int rows = pstmt.executeUpdate();

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

} catch (SQLException e) {

System.out.println("Insertion failed: " + e.getMessage());

}

}

}

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

ANS:

**import java.sql.Connection;**

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

public class FetchStudents {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/yourDatabase";

String user = "root";

String password = "yourPassword";

String query = "SELECT \* FROM students";

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

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

while (rs.next()) {

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

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

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

System.out.println("ID: " + id + ", Name: " + name + ", Marks: " + marks);

}

} catch (SQLException e) {

System.out.println("Error fetching data: " + e.getMessage());

}

}

}

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

ANS:

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 SearchStudentById {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/yourDatabase";

String user = "root";

String password = "yourPassword";

Scanner scanner = new Scanner(System.in);

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

int searchId = scanner.nextInt();

String query = "SELECT \* FROM students WHERE id = ?";

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

PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setInt(1, searchId);

ResultSet rs = pstmt.executeQuery();

if (rs.next()) {

System.out.println("Student Found:");

System.out.println("ID: " + rs.getInt("id"));

System.out.println("Name: " + rs.getString("name"));

System.out.println("Marks: " + rs.getInt("marks"));

} else {

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

}

rs.close();

} catch (SQLException e) {

System.out.println("Error during search: " + e.getMessage());

}

scanner.close();

}

}

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

ANS:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Scanner;

public class UpdateStudentDetails {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/mydb";

String user = "root";

String password = "Nara@123";

Scanner scanner = new Scanner(System.in);

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

int id = scanner.nextInt();

scanner.nextLine();

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

String newName = scanner.nextLine();

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

int newMarks = scanner.nextInt();

String updateSQL = "UPDATE students SET name = ?, marks = ? WHERE id = ?";

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

PreparedStatement pstmt = conn.prepareStatement(updateSQL)) {

pstmt.setString(1, newName);

pstmt.setInt(2, newMarks);

pstmt.setInt(3, id);

int rowsUpdated = pstmt.executeUpdate();

if (rowsUpdated > 0) {

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

} else {

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

}

} catch (SQLException e) {

System.out.println("Update failed: " + e.getMessage());

}

scanner.close();

}

}

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

ANS:

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.util.Scanner;

public class DeleteStudent {

public static void main(String[] args) {

String url = "jdbc:mysql://mydb";

String user = "root";

String password = "Nara@123”;

Scanner scanner = new Scanner(System.in);

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

int id = scanner.nextInt();

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

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

PreparedStatement pstmt = conn.prepareStatement(deleteSQL)) {

pstmt.setInt(1, id);

int rowsDeleted = pstmt.executeUpdate();

if (rowsDeleted > 0) {

System.out.println("Student deleted successfully.");

} else {

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

}

} catch (SQLException e) {

System.out.println("Deletion failed: " + e.getMessage());

}

scanner.close();

}

}

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

ANS:

import java.sql.\*;

import java.util.Scanner;

public class EmployeeCRUD {

private static final String URL = "jdbc:mysql://localhost:3306/mydb";

private static final String USER = "root";

private static final String PASSWORD = "Nara@123";

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

while (true) {

System.out.println("\n1. Create Employee");

System.out.println("2. Read All Employees");

System.out.println("3. Update Employee");

System.out.println("4. Delete Employee");

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

System.out.print("Choose option: ");

int choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1 -> createEmployee();

case 2 -> readEmployees();

case 3 -> updateEmployee();

case 4 -> deleteEmployee();

case 5 -> {

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

scanner.close();

return;

}

default -> System.out.println("Invalid choice");

}

}

}

private static void createEmployee() {

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

String name = scanner.nextLine();

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

String department = scanner.nextLine();

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

double salary = scanner.nextDouble();

scanner.nextLine();

String insertSQL = "INSERT INTO employee (name, department, salary) VALUES (?, ?, ?)";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

pstmt.setString(1, name);

pstmt.setString(2, department);

pstmt.setDouble(3, salary);

int rows = pstmt.executeUpdate();

System.out.println(rows > 0 ? "Employee added." : "Insertion failed.");

} catch (SQLException e) {

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

}

}

private static void readEmployees() {

String selectSQL = "SELECT \* FROM employee";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(selectSQL)) {

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

while (rs.next()) {

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

rs.getInt("id"),

rs.getString("name"),

rs.getString("department"),

rs.getDouble("salary"));

}

} catch (SQLException e) {

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

}

}

private static void updateEmployee() {

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

int id = scanner.nextInt();

scanner.nextLine();

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

String department = scanner.nextLine();

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

double salary = scanner.nextDouble();

scanner.nextLine();

String updateSQL = "UPDATE employee SET department = ?, salary = ? WHERE id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(updateSQL)) {

pstmt.setString(1, department);

pstmt.setDouble(2, salary);

pstmt.setInt(3, id);

int rows = pstmt.executeUpdate();

System.out.println(rows > 0 ? "Employee updated." : "No employee found with ID " + id);

} catch (SQLException e) {

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

}

}

private static void deleteEmployee() {

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

int id = scanner.nextInt();

scanner.nextLine();

String deleteSQL = "DELETE FROM employee WHERE id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(deleteSQL)) {

pstmt.setInt(1, id);

int rows = pstmt.executeUpdate();

System.out.println(rows > 0 ? "Employee deleted." : "No employee found with ID " + id);

} catch (SQLException e) {

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

}

}

}

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

ANS:

import java.sql.\*;

public class CountRowsInTable {

private static final String URL = "jdbc:mysql://localhost:3306/yourDatabase";

private static final String USER = "root";

private static final String PASSWORD = "yourPassword";

public static void main(String[] args) {

String query = "SELECT COUNT(\*) AS total FROM yourTable";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

if (rs.next()) {

int totalRows = rs.getInt("total");

System.out.println("Total number of rows: " + totalRows);

}

} catch (SQLException e) {

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

}

}

}

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

ANS:

import java.sql.\*;

import java.util.Scanner;

public class SortStudentsByName {

private static final String URL = "jdbc:mysql://localhost:3306/yourDatabase";

private static final String USER = "root";

private static final String PASSWORD = "yourPassword";

public static void main(String[] args) {

String query = "SELECT \* FROM students ORDER BY name ASC";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

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

while (rs.next()) {

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

rs.getInt("id"),

rs.getString("name"),

rs.getDouble("marks"));

}

} catch (SQLException e) {

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

}

}

}

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

ANS:

import java.sql.\*;

public class StudentsAbove75 {

private static final String URL = "jdbc:mysql://localhost:3306/yourDatabase";

private static final String USER = "root";

private static final String PASSWORD = "yourPassword";

public static void main(String[] args) {

String query = "SELECT \* FROM students WHERE percentage > 75";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(query)) {

System.out.println("Students with percentage greater than 75:");

while (rs.next()) {

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

rs.getInt("id"),

rs.getString("name"),

rs.getDouble("percentage"));

}

} catch (SQLException e) {

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

}

}

}

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

ANS:

import java.sql.\*;

public class InsertMultipleStudents {

private static final String URL = "jdbc:mysql://localhost:3306/mydb";

private static final String USER = "root";

private static final String PASSWORD = "Nara@123";

public static void main(String[] args) {

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

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {

// Sample data

Object[][] students = {

{1, "Alice", 85.5},

{2, "Bob", 78.0},

{3, "Charlie", 92.0}

};

for (Object[] student : students) {

pstmt.setInt(1, (int) student[0]);

pstmt.setString(2, (String) student[1]);

pstmt.setDouble(3, (double) student[2]);

pstmt.addBatch();

}

int[] results = pstmt.executeBatch();

System.out.println(results.length + " records inserted.");

} catch (SQLException e) {

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

}

}

}

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

ANS:  
import java.sql.\*;

public class TransactionExample {

private static final String URL = "jdbc:mysql://localhost:3306/yourDatabase";

private static final String USER = "root";

private static final String PASSWORD = "yourPassword";

public static void main(String[] args) {

String insertSQL = "INSERT INTO accounts (account\_id, balance) VALUES (?, ?)";

String updateSQL = "UPDATE accounts SET balance = balance - ? WHERE account\_id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement insertStmt = conn.prepareStatement(insertSQL);

PreparedStatement updateStmt = conn.prepareStatement(updateSQL)) {

conn.setAutoCommit(false);

insertStmt.setInt(1, 101);

insertStmt.setDouble(2, 5000);

insertStmt.executeUpdate();

updateStmt.setDouble(1, 6000);

updateStmt.setInt(2, 101);

int rows = updateStmt.executeUpdate();

if (rows == 0) {

conn.rollback();

System.out.println("Rollback: Insufficient balance or account not found.");

} else {

conn.commit();

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

}

} catch (SQLException e) {

System.out.println("Exception: " + e.getMessage());

}

}

}

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

ANS:  
import java.sql.\*;

public class JDBCExceptionHandling {

private static final String URL = "jdbc:mysql://localhost:3306/mydb";

private static final String USER = "root";

private static final String PASSWORD = "Nara@123";

public static void main(String[] args) {

String query = "SELECT \* FROM students WHERE id = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(query)) {

int studentId = -1; // Invalid ID example

pstmt.setInt(1, studentId);

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

if (rs.next()) {

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

rs.getInt("id"),

rs.getString("name"),

rs.getDouble("percentage"));

} else {

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

}

}

} catch (SQLException e) {

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

} catch (Exception e) {

System.out.println("Unexpected error: " + e.getMessage());

}

}

}

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

ANS:

import java.sql.\*;

import java.util.Scanner;

public class LoginSystem {

private static final String URL = "jdbc:mysql://localhost:3306/yourDatabase";

private static final String USER = "root";

private static final String PASSWORD = "yourPassword";

public static boolean verifyCredentials(String username, String password) {

String query = "SELECT \* FROM users WHERE username = ? AND password = ?";

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

PreparedStatement pstmt = conn.prepareStatement(query)) {

pstmt.setString(1, username);

pstmt.setString(2, password);

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

return rs.next();

}

} catch (SQLException e) {

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

return false;

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

String user = sc.nextLine();

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

String pass = sc.nextLine();

if (verifyCredentials(user, pass)) {

System.out.println("Login successful!");

} else {

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

}

sc.close();

}

}

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

ANS:

import java.sql.\*;

import java.util.Scanner;

public class StudentCRUD {

private static final String URL = "jdbc:mysql://localhost:3306/mydb";

private static final String USER = "root";

private static final String PASSWORD = "Nara@123";

public static void insertStudent(Scanner sc, Connection conn) throws SQLException {

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

int id = Integer.parseInt(sc.nextLine());

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

String name = sc.nextLine();

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

double marks = Double.parseDouble(sc.nextLine());

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

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

pstmt.setInt(1, id);

pstmt.setString(2, name);

pstmt.setDouble(3, marks);

int rows = pstmt.executeUpdate();

System.out.println(rows > 0 ? "Student inserted." : "Insertion failed.");

}

}

public static void searchStudent(Scanner sc, Connection conn) throws SQLException {

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

int id = Integer.parseInt(sc.nextLine());

String sql = "SELECT \* FROM students WHERE id = ?";

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

pstmt.setInt(1, id);

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

if (rs.next()) {

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

rs.getInt("id"),

rs.getString("name"),

rs.getDouble("marks"));

} else {

System.out.println("Student not found.");

}

}

}

}

public static void updateStudent(Scanner sc, Connection conn) throws SQLException {

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

int id = Integer.parseInt(sc.nextLine());

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

double marks = Double.parseDouble(sc.nextLine());

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

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

pstmt.setDouble(1, marks);

pstmt.setInt(2, id);

int rows = pstmt.executeUpdate();

System.out.println(rows > 0 ? "Student updated." : "Update failed or student not found.");

}

}

public static void main(String[] args) {

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

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

while (true) {

System.out.println("\nMenu:");

System.out.println("1. Insert Student");

System.out.println("2. Search Student");

System.out.println("3. Update Student");

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

System.out.print("Choose option: ");

String choice = sc.nextLine();

switch (choice) {

case "1" -> insertStudent(sc, conn);

case "2" -> searchStudent(sc, conn);

case "3" -> updateStudent(sc, conn);

case "4" -> {

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

return;

}

default -> System.out.println("Invalid option.");

}

}

} catch (SQLException e) {

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

}

}

}

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

ANS:MYSQL

CREATE TABLE Books (

book\_id INT PRIMARY KEY AUTO\_INCREMENT,

title VARCHAR(100),

author VARCHAR(100),

is\_issued BOOLEAN DEFAULT FALSE

);

CREATE TABLE Members (

member\_id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100)

);

CREATE TABLE IssuedBooks (

issue\_id INT PRIMARY KEY AUTO\_INCREMENT,

book\_id INT,

member\_id INT,

issue\_date DATE,

return\_date DATE,

FOREIGN KEY (book\_id) REFERENCES Books(book\_id),

FOREIGN KEY (member\_id) REFERENCES Members(member\_id)

);

JAVA CODE:

import java.sql.\*;

import java.time.LocalDate;

import java.util.Scanner;

public class LibraryManagement {

private static final String URL = "jdbc:mysql://localhost:3306/mydb";

private static final String USER = "root";

private static final String PASSWORD = "Nara@123";

public static void addBook(Connection conn, Scanner sc) throws SQLException {

System.out.print("Enter book title: ");

String title = sc.nextLine();

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

String author = sc.nextLine();

String sql = "INSERT INTO Books (title, author) VALUES (?, ?)";

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

pstmt.setString(1, title);

pstmt.setString(2, author);

int rows = pstmt.executeUpdate();

System.out.println(rows > 0 ? "Book added successfully." : "Failed to add book.");

}

}

public static void viewBooks(Connection conn) throws SQLException {

String sql = "SELECT \* FROM Books";

try (Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {

System.out.println("Books List:");

while (rs.next()) {

System.out.printf("ID: %d, Title: %s, Author: %s, Issued: %b%n",

rs.getInt("book\_id"),

rs.getString("title"),

rs.getString("author"),

rs.getBoolean("is\_issued"));

}

}

}

public static void issueBook(Connection conn, Scanner sc) throws SQLException {

System.out.print("Enter Book ID to issue: ");

int bookId = Integer.parseInt(sc.nextLine());

String checkSql = "SELECT is\_issued FROM Books WHERE book\_id = ?";

try (PreparedStatement checkStmt = conn.prepareStatement(checkSql)) {

checkStmt.setInt(1, bookId);

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

if (!rs.next()) {

System.out.println("Book not found.");

return;

}

if (rs.getBoolean("is\_issued")) {

System.out.println("Book already issued.");

return;

}

}

}

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

int memberId = Integer.parseInt(sc.nextLine());

String issueSql = "INSERT INTO IssuedBooks (book\_id, member\_id, issue\_date) VALUES (?, ?, ?)";

String updateBookSql = "UPDATE Books SET is\_issued = TRUE WHERE book\_id = ?";

try (PreparedStatement issueStmt = conn.prepareStatement(issueSql);

PreparedStatement updateStmt = conn.prepareStatement(updateBookSql)) {

conn.setAutoCommit(false);

issueStmt.setInt(1, bookId);

issueStmt.setInt(2, memberId);

issueStmt.setDate(3, Date.valueOf(LocalDate.now()));

issueStmt.executeUpdate();

updateStmt.setInt(1, bookId);

updateStmt.executeUpdate();

conn.commit();

System.out.println("Book issued successfully.");

} catch (SQLException e) {

conn.rollback();

System.out.println("Failed to issue book: " + e.getMessage());

} finally {

conn.setAutoCommit(true);

}

}

public static void returnBook(Connection conn, Scanner sc) throws SQLException {

System.out.print("Enter Book ID to return: ");

int bookId = Integer.parseInt(sc.nextLine());

String checkSql = "SELECT issue\_id FROM IssuedBooks WHERE book\_id = ? AND return\_date IS NULL";

try (PreparedStatement checkStmt = conn.prepareStatement(checkSql)) {

checkStmt.setInt(1, bookId);

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

if (!rs.next()) {

System.out.println("No active issue record found for this book.");

return;

}

int issueId = rs.getInt("issue\_id");

String updateIssueSql = "UPDATE IssuedBooks SET return\_date = ? WHERE issue\_id = ?";

String updateBookSql = "UPDATE Books SET is\_issued = FALSE WHERE book\_id = ?";

try (PreparedStatement updateIssueStmt = conn.prepareStatement(updateIssueSql);

PreparedStatement updateBookStmt = conn.prepareStatement(updateBookSql)) {

conn.setAutoCommit(false);

updateIssueStmt.setDate(1, Date.valueOf(LocalDate.now()));

updateIssueStmt.setInt(2, issueId);

updateIssueStmt.executeUpdate();

updateBookStmt.setInt(1, bookId);

updateBookStmt.executeUpdate();

conn.commit();

System.out.println("Book returned successfully.");

} catch (SQLException e) {

conn.rollback();

System.out.println("Failed to return book: " + e.getMessage());

} finally {

conn.setAutoCommit(true);

}

}

}

}

public static void main(String[] args) {

try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD);

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

while (true) {

System.out.println("\nLibrary Menu:");

System.out.println("1. Add Book");

System.out.println("2. View All Books");

System.out.println("3. Issue Book");

System.out.println("4. Return Book");

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

System.out.print("Choose an option: ");

String choice = sc.nextLine();

switch (choice) {

case "1" -> addBook(conn, sc);

case "2" -> viewBooks(conn);

case "3" -> issueBook(conn, sc);

case "4" -> returnBook(conn, sc);

case "5" -> {

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

return;

}

default -> System.out.println("Invalid choice");

}

}

} catch (SQLException e) {

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

}

}

}

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

* Register new patient
* Assign doctor
* Generate billing

ANS:

CREATE TABLE Patients (

patient\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

age INT,

gender VARCHAR(10)

);

CREATE TABLE Doctors (

doctor\_id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

specialty VARCHAR(100)

);

CREATE TABLE PatientDoctor (

id INT AUTO\_INCREMENT PRIMARY KEY,

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

ANS:  
import java.sql.\*;

import java.io.\*;

public class ReportGenerator {

public static void main(String[] args) {

String url = "jdbc:mysql://localhost:3306/mydb";

String user = "root";

String password = "Nara@123";

String query = "SELECT \* FROM your\_table"; // Replace with your table

try (

Connection con = DriverManager.getConnection(url, user, password);

Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery(query);

BufferedWriter bw = new BufferedWriter(new FileWriter("report.csv"));

) {

ResultSetMetaData rsmd = rs.getMetaData();

int columns = rsmd.getColumnCount();

// Write header row

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

bw.write(rsmd.getColumnName(i));

if (i < columns) bw.write(",");

}

bw.newLine();

// Write data rows

while (rs.next()) {

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

String data = rs.getString(i);

if (data != null && data.contains(",")) {

data = "\"" + data + "\""; // Handle commas in data

}

bw.write(data != null ? data : "");

if (i < columns) bw.write(",");

}

bw.newLine();

}

System.out.println("Report generated successfully.");

} catch (SQLException | IOException e) {

e.printStackTrace();

}

}

}