# Day – 11

## JDBC

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

Code:

import java.sql.\*;  
  
public class MySQLConnection {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try {  
 Connection connection = DriverManager.getConnection(url, username, password);  
 System.out.println("Connected to MySQL database!");  
 connection.close();  
 } catch (SQLException e) {  
 System.out.println("Connection failed!");  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
  
public class InsertStudent {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "INSERT INTO students (id, name, percentage) VALUES (?, ?, ?)";  
 PreparedStatement statement = conn.prepareStatement(sql);  
   
 statement.setInt(1, 101);  
 statement.setString(2, "Alice");  
 statement.setDouble(3, 85.5);  
 statement.executeUpdate();  
   
 System.out.println("Student record inserted successfully.");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
  
public class FetchStudents {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT \* FROM students");  
   
 System.out.println("Student Records:");  
 while (rs.next()) {  
 System.out.println("ID: " + rs.getInt("id") +   
 ", Name: " + rs.getString("name") +   
 ", Percentage: " + rs.getDouble("percentage"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class SearchStudent {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter student ID: ");  
 int id = scanner.nextInt();  
   
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "SELECT \* FROM students WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 stmt.setInt(1, id);  
   
 ResultSet rs = stmt.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("Percentage: " + rs.getDouble("percentage"));  
 } else {  
 System.out.println("Student not found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class UpdateStudent {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter student ID to update: ");  
 int id = scanner.nextInt();  
 scanner.nextLine(); // consume newline  
   
 System.out.print("Enter new name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter new percentage: ");  
 double percentage = scanner.nextDouble();  
   
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "UPDATE students SET name = ?, percentage = ? WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setString(1, name);  
 stmt.setDouble(2, percentage);  
 stmt.setInt(3, id);  
   
 int rowsAffected = stmt.executeUpdate();  
 if (rowsAffected > 0) {  
 System.out.println("Student updated successfully.");  
 } else {  
 System.out.println("No student found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class DeleteStudent {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter student ID to delete: ");  
 int id = scanner.nextInt();  
   
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "DELETE FROM students WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 stmt.setInt(1, id);  
   
 int rowsAffected = stmt.executeUpdate();  
 if (rowsAffected > 0) {  
 System.out.println("Student deleted successfully.");  
 } else {  
 System.out.println("No student found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class EmployeeCRUD {  
 private static final String URL = "jdbc:mysql://localhost:3306/company";  
 private static final String USERNAME = "root";  
 private static final String PASSWORD = "password";  
   
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
   
 while (true) {  
 System.out.println("\nEmployee CRUD Operations");  
 System.out.println("1. Create Employee");  
 System.out.println("2. Read Employees");  
 System.out.println("3. Update Employee");  
 System.out.println("4. Delete Employee");  
 System.out.println("5. Exit");  
 System.out.print("Enter choice: ");  
   
 int choice = scanner.nextInt();  
 scanner.nextLine(); // consume newline  
   
 switch (choice) {  
 case 1:  
 createEmployee(scanner);  
 break;  
 case 2:  
 readEmployees();  
 break;  
 case 3:  
 updateEmployee(scanner);  
 break;  
 case 4:  
 deleteEmployee(scanner);  
 break;  
 case 5:  
 System.exit(0);  
 default:  
 System.out.println("Invalid choice!");  
 }  
 }  
 }  
   
 private static void createEmployee(Scanner scanner) {  
 System.out.print("Enter employee ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter employee name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter employee salary: ");  
 double salary = scanner.nextDouble();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "INSERT INTO employees (id, name, salary) VALUES (?, ?, ?)";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setInt(1, id);  
 stmt.setString(2, name);  
 stmt.setDouble(3, salary);  
   
 stmt.executeUpdate();  
 System.out.println("Employee created successfully.");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
   
 private static void readEmployees() {  
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT \* FROM employees");  
   
 System.out.println("\nEmployee List:");  
 while (rs.next()) {  
 System.out.println("ID: " + rs.getInt("id") +   
 ", Name: " + rs.getString("name") +   
 ", Salary: " + rs.getDouble("salary"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
   
 private static void updateEmployee(Scanner scanner) {  
 System.out.print("Enter employee ID to update: ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter new name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter new salary: ");  
 double salary = scanner.nextDouble();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "UPDATE employees SET name = ?, salary = ? WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setString(1, name);  
 stmt.setDouble(2, salary);  
 stmt.setInt(3, id);  
   
 int rows = stmt.executeUpdate();  
 if (rows > 0) {  
 System.out.println("Employee updated successfully.");  
 } else {  
 System.out.println("Employee not found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
   
 private static void deleteEmployee(Scanner scanner) {  
 System.out.print("Enter employee ID to delete: ");  
 int id = scanner.nextInt();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "DELETE FROM employees WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 stmt.setInt(1, id);  
   
 int rows = stmt.executeUpdate();  
 if (rows > 0) {  
 System.out.println("Employee deleted successfully.");  
 } else {  
 System.out.println("Employee not found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
  
public class CountRows {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT COUNT(\*) AS total FROM students");  
   
 if (rs.next()) {  
 System.out.println("Total students: " + rs.getInt("total"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
  
public class SortStudentsByName {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT \* FROM students ORDER BY name ASC");  
   
 System.out.println("Students sorted by name:");  
 while (rs.next()) {  
 System.out.println("ID: " + rs.getInt("id") +   
 ", Name: " + rs.getString("name") +   
 ", Percentage: " + rs.getDouble("percentage"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
  
public class FilterStudentsByPercentage {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "SELECT \* FROM students WHERE percentage > 75";  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(sql);  
   
 System.out.println("Students with >75%:");  
 while (rs.next()) {  
 System.out.println("ID: " + rs.getInt("id") +   
 ", Name: " + rs.getString("name") +   
 ", Percentage: " + rs.getDouble("percentage"));  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class InsertMultipleStudents {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "INSERT INTO students (id, name, percentage) VALUES (?, ?, ?)";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 Scanner scanner = new Scanner(System.in);  
 char choice;  
   
 do {  
 System.out.print("Enter student ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter student name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter student percentage: ");  
 double percentage = scanner.nextDouble();  
   
 stmt.setInt(1, id);  
 stmt.setString(2, name);  
 stmt.setDouble(3, percentage);  
 stmt.addBatch();  
   
 System.out.print("Add another student? (y/n): ");  
 choice = scanner.next().charAt(0);  
 scanner.nextLine();  
 } while (choice == 'y' || choice == 'Y');  
   
 int[] result = stmt.executeBatch();  
 System.out.println(result.length + " students added successfully.");  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class TransactionManagement {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/bank";  
 String username = "root";  
 String password = "password";  
   
 Connection conn = null;  
 try {  
 conn = DriverManager.getConnection(url, username, password);  
 conn.setAutoCommit(false); // Start transaction  
   
 // Transfer 100 from account 1 to account 2  
 String debit = "UPDATE accounts SET balance = balance - 100 WHERE id = 1";  
 String credit = "UPDATE accounts SET balance = balance + 100 WHERE id = 2";  
   
 Statement stmt = conn.createStatement();  
 stmt.executeUpdate(debit);  
 stmt.executeUpdate(credit);  
   
 // Verify balances  
 ResultSet rs = stmt.executeQuery("SELECT balance FROM accounts WHERE id = 1");  
 rs.next();  
 double balance = rs.getDouble(1);  
   
 if (balance < 0) {  
 conn.rollback();  
 System.out.println("Transaction rolled back due to insufficient funds.");  
 } else {  
 conn.commit();  
 System.out.println("Transaction committed successfully.");  
 }  
 } catch (SQLException e) {  
 try {  
 if (conn != null) {  
 conn.rollback();  
 }  
 } catch (SQLException ex) {  
 ex.printStackTrace();  
 }  
 e.printStackTrace();  
 } finally {  
 try {  
 if (conn != null) {  
 conn.setAutoCommit(true);  
 conn.close();  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class JDBCExceptionHandling {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter student ID: ");  
 int id = scanner.nextInt();  
   
 String sql = "SELECT \* FROM students WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 stmt.setInt(1, id);  
   
 ResultSet rs = stmt.executeQuery();  
 if (rs.next()) {  
 System.out.println("Student found: " + rs.getString("name"));  
 } else {  
 System.out.println("No student found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 System.err.println("Database error occurred:");  
 System.err.println("Error code: " + e.getErrorCode());  
 System.err.println("SQL state: " + e.getSQLState());  
 System.err.println("Message: " + e.getMessage());  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.util.Scanner;  
  
public class LoginSystem {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/auth";  
 String username = "root";  
 String password = "password";  
   
 Scanner scanner = new Scanner(System.in);  
 System.out.print("Enter username: ");  
 String inputUser = scanner.nextLine();  
 System.out.print("Enter password: ");  
 String inputPass = scanner.nextLine();  
   
 try (Connection conn = DriverManager.getConnection(url, username, password)) {  
 String sql = "SELECT \* FROM users WHERE username = ? AND password = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setString(1, inputUser);  
 stmt.setString(2, inputPass);  
   
 ResultSet rs = stmt.executeQuery();  
 if (rs.next()) {  
 System.out.println("Login successful! Welcome " + rs.getString("name"));  
 } else {  
 System.out.println("Invalid username or password.");  
 }  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
}

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

Code:  
import java.sql.\*;  
import java.util.Scanner;  
  
public class MenuDrivenDBApp {  
 private static final String URL = "jdbc:mysql://localhost:3306/school";  
 private static final String USERNAME = "root";  
 private static final String PASSWORD = "password";  
   
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
   
 while (true) {  
 System.out.println("\nStudent Database Menu");  
 System.out.println("1. Add Student");  
 System.out.println("2. View Student");  
 System.out.println("3. Update Student");  
 System.out.println("4. Delete Student");  
 System.out.println("5. Exit");  
 System.out.print("Enter choice: ");  
   
 int choice = scanner.nextInt();  
 scanner.nextLine(); // consume newline  
   
 switch (choice) {  
 case 1:  
 addStudent(scanner);  
 break;  
 case 2:  
 viewStudent(scanner);  
 break;  
 case 3:  
 updateStudent(scanner);  
 break;  
 case 4:  
 deleteStudent(scanner);  
 break;  
 case 5:  
 System.exit(0);  
 default:  
 System.out.println("Invalid choice!");  
 }  
 }  
 }  
   
 private static void addStudent(Scanner scanner) {  
 System.out.print("Enter student ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter student name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter student percentage: ");  
 double percentage = scanner.nextDouble();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "INSERT INTO students (id, name, percentage) VALUES (?, ?, ?)";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setInt(1, id);  
 stmt.setString(2, name);  
 stmt.setDouble(3, percentage);  
   
 stmt.executeUpdate();  
 System.out.println("Student added successfully.");  
 } catch (SQLException e) {  
 System.out.println("Error adding student: " + e.getMessage());  
 }  
 }  
   
 private static void viewStudent(Scanner scanner) {  
 System.out.print("Enter student ID: ");  
 int id = scanner.nextInt();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "SELECT \* FROM students WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 stmt.setInt(1, id);  
   
 ResultSet rs = stmt.executeQuery();  
 if (rs.next()) {  
 System.out.println("Student Details:");  
 System.out.println("ID: " + rs.getInt("id"));  
 System.out.println("Name: " + rs.getString("name"));  
 System.out.println("Percentage: " + rs.getDouble("percentage"));  
 } else {  
 System.out.println("Student not found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 System.out.println("Error viewing student: " + e.getMessage());  
 }  
 }  
   
 private static void updateStudent(Scanner scanner) {  
 System.out.print("Enter student ID to update: ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter new name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter new percentage: ");  
 double percentage = scanner.nextDouble();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "UPDATE students SET name = ?, percentage = ? WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setString(1, name);  
 stmt.setDouble(2, percentage);  
 stmt.setInt(3, id);  
   
 int rows = stmt.executeUpdate();  
 if (rows > 0) {  
 System.out.println("Student updated successfully.");  
 } else {  
 System.out.println("Student not found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 System.out.println("Error updating student: " + e.getMessage());  
 }  
 }  
   
 private static void deleteStudent(Scanner scanner) {  
 System.out.print("Enter student ID to delete: ");  
 int id = scanner.nextInt();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "DELETE FROM students WHERE id = ?";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
 stmt.setInt(1, id);  
   
 int rows = stmt.executeUpdate();  
 if (rows > 0) {  
 System.out.println("Student deleted successfully.");  
 } else {  
 System.out.println("Student not found with ID: " + id);  
 }  
 } catch (SQLException e) {  
 System.out.println("Error deleting student: " + e.getMessage());  
 }  
 }  
}

16. 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

Code:  
import java.sql.\*;  
import java.util.Scanner;  
  
public class LibraryManagement {  
 private static final String URL = "jdbc:mysql://localhost:3306/library";  
 private static final String USERNAME = "root";  
 private static final String PASSWORD = "password";  
   
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
   
 while (true) {  
 System.out.println("\nLibrary Management System");  
 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("Enter choice: ");  
   
 int choice = scanner.nextInt();  
 scanner.nextLine(); // consume newline  
   
 switch (choice) {  
 case 1:  
 addBook(scanner);  
 break;  
 case 2:  
 viewBooks();  
 break;  
 case 3:  
 issueBook(scanner);  
 break;  
 case 4:  
 returnBook(scanner);  
 break;  
 case 5:  
 System.exit(0);  
 default:  
 System.out.println("Invalid choice!");  
 }  
 }  
 }  
   
 private static void addBook(Scanner scanner) {  
 System.out.print("Enter book ID: ");  
 int id = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter book title: ");  
 String title = scanner.nextLine();  
   
 System.out.print("Enter author: ");  
 String author = scanner.nextLine();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "INSERT INTO books (book\_id, title, author, available) VALUES (?, ?, ?, ?)";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setInt(1, id);  
 stmt.setString(2, title);  
 stmt.setString(3, author);  
 stmt.setBoolean(4, true);  
   
 stmt.executeUpdate();  
 System.out.println("Book added successfully.");  
 } catch (SQLException e) {  
 System.out.println("Error adding book: " + e.getMessage());  
 }  
 }  
   
 private static void viewBooks() {  
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery("SELECT \* FROM books");  
   
 System.out.println("\nBook List:");  
 while (rs.next()) {  
 System.out.println("ID: " + rs.getInt("book\_id") +   
 ", Title: " + rs.getString("title") +   
 ", Author: " + rs.getString("author") +   
 ", Available: " + rs.getBoolean("available"));  
 }  
 } catch (SQLException e) {  
 System.out.println("Error viewing books: " + e.getMessage());  
 }  
 }  
   
 private static void issueBook(Scanner scanner) {  
 System.out.print("Enter book ID to issue: ");  
 int bookId = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter member ID: ");  
 int memberId = scanner.nextInt();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 conn.setAutoCommit(false);  
   
 // Check if book is available  
 String checkSql = "SELECT available FROM books WHERE book\_id = ? FOR UPDATE";  
 PreparedStatement checkStmt = conn.prepareStatement(checkSql);  
 checkStmt.setInt(1, bookId);  
 ResultSet rs = checkStmt.executeQuery();  
   
 if (rs.next() && rs.getBoolean("available")) {  
 // Update book status  
 String updateBookSql = "UPDATE books SET available = false WHERE book\_id = ?";  
 PreparedStatement updateBookStmt = conn.prepareStatement(updateBookSql);  
 updateBookStmt.setInt(1, bookId);  
 updateBookStmt.executeUpdate();  
   
 // Create issue record  
 String issueSql = "INSERT INTO book\_issues (book\_id, member\_id, issue\_date) VALUES (?, ?, CURDATE())";  
 PreparedStatement issueStmt = conn.prepareStatement(issueSql);  
 issueStmt.setInt(1, bookId);  
 issueStmt.setInt(2, memberId);  
 issueStmt.executeUpdate();  
   
 conn.commit();  
 System.out.println("Book issued successfully.");  
 } else {  
 conn.rollback();  
 System.out.println("Book not available for issue.");  
 }  
 } catch (SQLException e) {  
 System.out.println("Error issuing book: " + e.getMessage());  
 }  
 }  
   
 private static void returnBook(Scanner scanner) {  
 System.out.print("Enter book ID to return: ");  
 int bookId = scanner.nextInt();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 conn.setAutoCommit(false);  
   
 // Update book status  
 String updateBookSql = "UPDATE books SET available = true WHERE book\_id = ?";  
 PreparedStatement updateBookStmt = conn.prepareStatement(updateBookSql);  
 updateBookStmt.setInt(1, bookId);  
 int rowsUpdated = updateBookStmt.executeUpdate();  
   
 if (rowsUpdated > 0) {  
 // Update issue record  
 String returnSql = "UPDATE book\_issues SET return\_date = CURDATE() WHERE book\_id = ? AND return\_date IS NULL";  
 PreparedStatement returnStmt = conn.prepareStatement(returnSql);  
 returnStmt.setInt(1, bookId);  
 returnStmt.executeUpdate();  
   
 conn.commit();  
 System.out.println("Book returned successfully.");  
 } else {  
 conn.rollback();  
 System.out.println("Invalid book ID or book not issued.");  
 }  
 } catch (SQLException e) {  
 System.out.println("Error returning book: " + e.getMessage());  
 }  
 }  
}

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

Register new patient

Assign doctor

Generate billing

Code:  
import java.sql.\*;  
import java.util.Scanner;  
  
public class HospitalManagement {  
 private static final String URL = "jdbc:mysql://localhost:3306/hospital";  
 private static final String USERNAME = "root";  
 private static final String PASSWORD = "password";  
   
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
   
 while (true) {  
 System.out.println("\nHospital Management System");  
 System.out.println("1. Register Patient");  
 System.out.println("2. Assign Doctor");  
 System.out.println("3. Generate Bill");  
 System.out.println("4. Exit");  
 System.out.print("Enter choice: ");  
   
 int choice = scanner.nextInt();  
 scanner.nextLine(); // consume newline  
   
 switch (choice) {  
 case 1:  
 registerPatient(scanner);  
 break;  
 case 2:  
 assignDoctor(scanner);  
 break;  
 case 3:  
 generateBill(scanner);  
 break;  
 case 4:  
 System.exit(0);  
 default:  
 System.out.println("Invalid choice!");  
 }  
 }  
 }  
   
 private static void registerPatient(Scanner scanner) {  
 System.out.print("Enter patient name: ");  
 String name = scanner.nextLine();  
   
 System.out.print("Enter patient age: ");  
 int age = scanner.nextInt();  
 scanner.nextLine();  
   
 System.out.print("Enter patient gender (M/F): ");  
 String gender = scanner.nextLine();  
   
 System.out.print("Enter patient contact: ");  
 String contact = scanner.nextLine();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "INSERT INTO patients (name, age, gender, contact) VALUES (?, ?, ?, ?)";  
 PreparedStatement stmt = conn.prepareStatement(sql, Statement.RETURN\_GENERATED\_KEYS);  
   
 stmt.setString(1, name);  
 stmt.setInt(2, age);  
 stmt.setString(3, gender);  
 stmt.setString(4, contact);  
   
 stmt.executeUpdate();  
   
 ResultSet rs = stmt.getGeneratedKeys();  
 if (rs.next()) {  
 System.out.println("Patient registered successfully. ID: " + rs.getInt(1));  
 }  
 } catch (SQLException e) {  
 System.out.println("Error registering patient: " + e.getMessage());  
 }  
 }  
   
 private static void assignDoctor(Scanner scanner) {  
 System.out.print("Enter patient ID: ");  
 int patientId = scanner.nextInt();  
   
 System.out.print("Enter doctor ID: ");  
 int doctorId = scanner.nextInt();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "INSERT INTO patient\_doctor (patient\_id, doctor\_id) VALUES (?, ?)";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setInt(1, patientId);  
 stmt.setInt(2, doctorId);  
   
 int rows = stmt.executeUpdate();  
 if (rows > 0) {  
 System.out.println("Doctor assigned successfully.");  
 } else {  
 System.out.println("Failed to assign doctor.");  
 }  
 } catch (SQLException e) {  
 System.out.println("Error assigning doctor: " + e.getMessage());  
 }  
 }  
   
 private static void generateBill(Scanner scanner) {  
 System.out.print("Enter patient ID: ");  
 int patientId = scanner.nextInt();  
   
 System.out.print("Enter amount: ");  
 double amount = scanner.nextDouble();  
   
 System.out.print("Enter description: ");  
 scanner.nextLine(); // consume newline  
 String description = scanner.nextLine();  
   
 try (Connection conn = DriverManager.getConnection(URL, USERNAME, PASSWORD)) {  
 String sql = "INSERT INTO bills (patient\_id, amount, description, bill\_date) VALUES (?, ?, ?, CURDATE())";  
 PreparedStatement stmt = conn.prepareStatement(sql);  
   
 stmt.setInt(1, patientId);  
 stmt.setDouble(2, amount);  
 stmt.setString(3, description);  
   
 stmt.executeUpdate();  
 System.out.println("Bill generated successfully.");  
 } catch (SQLException e) {  
 System.out.println("Error generating bill: " + e.getMessage());  
 }  
 }  
}

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

Code:

import java.sql.\*;  
import java.io.\*;  
  
public class ReportGenerator {  
 public static void main(String[] args) {  
 String url = "jdbc:mysql://localhost:3306/school";  
 String username = "root";  
 String password = "password";  
 String outputFile = "students\_report.csv";  
   
 try (Connection conn = DriverManager.getConnection(url, username, password);  
 Statement stmt = conn.createStatement();  
 PrintWriter writer = new PrintWriter(new FileWriter(outputFile))) {  
   
 ResultSet rs = stmt.executeQuery("SELECT \* FROM students");  
 ResultSetMetaData meta = rs.getMetaData();  
 int colCount = meta.getColumnCount();  
   
 // Write CSV header  
 for (int i = 1; i <= colCount; i++) {  
 writer.print(meta.getColumnName(i));  
 if (i < colCount) writer.print(",");  
 }  
 writer.println();  
   
 // Write data rows  
 while (rs.next()) {  
 for (int i = 1; i <= colCount; i++) {  
 writer.print(rs.getString(i));  
 if (i < colCount) writer.print(",");  
 }  
 writer.println();  
 }  
   
 System.out.println("Report generated successfully: " + outputFile);  
 } catch (SQLException | IOException e) {  
 e.printStackTrace();  
 }  
 }  
}