□ JDBC Exercises

1. **Exercise 1: Establish JDBC Connection**
   - Write a Java program to connect to a MySQL database named `testdb` using JDBC.
2. **Exercise 2: Create a Table**
   - Create a table `Students` with the following columns:
     - `id` (INT, primary key)
     - `name` (VARCHAR)
     - `email` (VARCHAR)
3. **Exercise 3: Insert Data into Table**
   - Insert 2 student records using `Statement`.
4. **Exercise 4: Retrieve and Display Records**
   - Display all records from the `Students` table using `ResultSet`.

---

5. **Exercise 5: Insert Data Using `PreparedStatement`**
   - Use `PreparedStatement` to insert data into `Students` table.
6. **Exercise 6: Update a Record**
   - Update a student’s email based on the student `id`.
7. **Exercise 7: Delete a Record**
   - Delete a student from the table using `PreparedStatement`.
8. **Exercise 8: Search by Name**
   - Prompt user for a name and fetch all matching student records.

---

9. **Exercise 9: Use Stored Procedure**
   - Create a stored procedure `getStudentById` and invoke it using `CallableStatement`.
10. **Exercise 10: Create a Mini Project**

- Create a CLI Java app to manage students:
  - Add student
  - Update student
  - Delete student
  - View all students

Solutions & Explanations

Exercise 1: Establish JDBC Connection

```
import java.sql.*;

public class DBConnection {
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection(
                "jdbc:mysql://localhost:3306/testdb", "root", "password"
            );
            System.out.println("Connected!");
            conn.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Explanation: Loads the MySQL driver implicitly, connects to DB using URL, user, and password.

Exercise 2: Create Table

```
Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb", "root",
"password");
Statement stmt = conn.createStatement();
String createSQL = "CREATE TABLE Students (id INT PRIMARY KEY, name
VARCHAR(100), email VARCHAR(100))";
stmt.executeUpdate(createSQL);
System.out.println("Table created successfully.");
conn.close();
```

Explanation: Simple DDL command via Statement.

□ Exercise 3: Insert Data Using Statement

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("INSERT INTO Students VALUES (1, 'Alice',
'alice@example.com')");
stmt.executeUpdate("INSERT INTO Students VALUES (2, 'Bob',
'bob@example.com')");
System.out.println("Data inserted.");
```

□ Exercise 4: Retrieve Records

```
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Students");

while (rs.next()) {
    System.out.println(rs.getInt("id") + " - " + rs.getString("name") + " - "
+ rs.getString("email"));
}
```

□ Exercise 5: Insert with PreparedStatement

```
String sql = "INSERT INTO Students (id, name, email) VALUES (?, ?, ?)";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, 3);
pstmt.setString(2, "Charlie");
pstmt.setString(3, "charlie@example.com");
pstmt.executeUpdate();
```

Use this method to prevent SQL injection.

□ Exercise 6: Update a Record

```
String sql = "UPDATE Students SET email = ? WHERE id = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, "newbob@example.com");
pstmt.setInt(2, 2);
pstmt.executeUpdate();
```

□ Exercise 7: Delete a Record

```
String sql = "DELETE FROM Students WHERE id = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setInt(1, 1);
pstmt.executeUpdate();
```

□ Exercise 8: Search by Name

```
Scanner sc = new Scanner(System.in);
System.out.print("Enter name to search: ");
String name = sc.nextLine();

String sql = "SELECT * FROM Students WHERE name = ?";
PreparedStatement pstmt = conn.prepareStatement(sql);
pstmt.setString(1, name);
ResultSet rs = pstmt.executeQuery();

while (rs.next()) {
    System.out.println(rs.getInt("id") + " - " + rs.getString("name") + " - "
+ rs.getString("email"));
}
```

□ Exercise 9: Stored Procedure

In MySQL:

```
DELIMITER //
CREATE PROCEDURE getStudentById(IN stu_id INT)
BEGIN
  SELECT * FROM Students WHERE id = stu_id;
END //
DELIMITER ;
```

Java Code:

```
CallableStatement cs = conn.prepareCall("{call getStudentById(?)}");
cs.setInt(1, 2);
ResultSet rs = cs.executeQuery();

while (rs.next()) {
    System.out.println("Name: " + rs.getString("name") + ", Email: " +
rs.getString("email"));
}
```

□ Exercise 10: Mini Project – CLI Student Manager

```
public class StudentManager {
    public static void main(String[] args) throws Exception {
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/testdb", "root",
"password");
        Scanner sc = new Scanner(System.in);
        int choice;

        do {
            System.out.println("1. Add\n2. View\n3. Update\n4. Delete\n5.
Exit");
            choice = sc.nextInt();
            sc.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    System.out.print("ID: "); int id = sc.nextInt();
                    sc.nextLine();
                    System.out.print("Name: "); String name = sc.nextLine();
                    System.out.print("Email: "); String email =
sc.nextLine();
                    PreparedStatement ps = conn.prepareStatement("INSERT INTO
Students VALUES (?, ?, ?)");
                    ps.setInt(1, id); ps.setString(2, name); ps.setString(3,
email);
                    ps.executeUpdate();
                    break;

                case 2:
                    ResultSet rs =
conn.createStatement().executeQuery("SELECT * FROM Students");
                    while (rs.next()) {
                        System.out.println(rs.getInt("id") + " - " +
rs.getString("name") + " - " + rs.getString("email"));
                    }
                    break;

                case 3:
                    System.out.print("Enter ID to update: "); int uid =
sc.nextInt();
                    sc.nextLine();
                    System.out.print("New Email: "); String newEmail =
sc.nextLine();
                    PreparedStatement ups = conn.prepareStatement("UPDATE
Students SET email=? WHERE id=?");
                    ups.setString(1, newEmail); ups.setInt(2, uid);
ups.executeUpdate();
                    break;

                case 4:
                    System.out.print("Enter ID to delete: "); int did =
sc.nextInt();
                    PreparedStatement dps = conn.prepareStatement("DELETE
FROM Students WHERE id=?");
```

```
                    dps.setInt(1, did); dps.executeUpdate();
                    break;

                case 5:
                    System.out.println("Exiting...");
                    break;
            }
        } while (choice != 5);

        conn.close();
    }
}
```

□ Summary

Exercise	Topic	Purpose
1	DB Connection	Basic JDBC setup
2	Create Table	DDL using Statement
3	Insert	Insert with Statement
4	Fetch	Read with ResultSet
5	PreparedStatement	Secure insert
6	Update	Modify data
7	Delete	Remove data
8	Search	Dynamic query
9	Stored Procedure	Call DB logic
10	Mini Project	Combine all in real app