Connected Architecture

Task-1

Public void GetTransactions(d1 DateTime , d2 DateTime)
{
// logic to display all records from Employees who date of join between 2 dates using procedures
}

Task-2

Public void GetCommonRecords(int id)
{
// logic to display common records from Employee and department based on id
}

Task-3

Public void InsertRecordsusingtrans()
{
// logic to insert records to employee and department using transaction
}

Task-4

Connected Insert + Fetch New Identity

For a **Employee** table with an identity primary key:

- Insert a new Employee using INSERT command.
- Immediately fetch the newly inserted identity using:
  - SCOPE_IDENTITY()
- Validate that the ID exists by fetching it again with a new command.

Task-5

Multi-Result Reader with Joins

A company has **Employees** and **Department** tables.
Write a C# program that:

- Executes a single **SqlCommand** that returns **two result sets**:
    1. List of employees
    2. List of Departments
- Reads first result set via SqlDataReader.Read(), then moves to the next using NextResult().

Task-6

Using Stored Procedure That Returns Multiple Output Parameters

Stored procedure:

```
sp_GetEmployeeDet
  @Empid,
  @DateofJoin OUTPUT,
  @Department  OUTPUT
```

Task:

- Call this stored procedure using SqlCommand in connected mode.
- Retrieve output parameters.
- Display formatted summary.

- **Check if connection is opened or not, display appropriate message if connection is not opened**
- **Handle exception for all methods using SqlException Class**
- **Pass all the data dynamically.**
- **Using single connection for all above operations**