Part 1 : Analyze Legacy Code

```
public DataTable GetCustomerInfo(string id)
{
    var dt = new DataTable();
    using (var conn = new SqlConnection("...")) // Connection string is hardcoded
    {
        conn.Open();
        var sql = "SELECT * FROM Customer WHERE id = '" + id + "'";
        using (var da = new SqlDataAdapter(sql, conn))
        {
            da.Fill(dt);
        }
    }
    return dt;
}
```

TASK

1. In your own words, what is the primary purpose of this function?
   Ans. the primary purpose of this GetCustomerInfo function is to retrieve customer details from a database based on a provided customer ID. It aims to fetch all columns for a matching customer record and return them in a DataTable object.
2. Identify at least three distinct problems with this implementation. Consider aspects such as security, maintainability, and performance.
   Ans. Security : the SQL query string ("SELECT * FROM Customer WHERE id = '" + id + "'") without any sanitization or parameterization. This creates a severe SQL Injection vulnerability
   Maintainability: The connection string is directly embedded ("hardcoded") within the function's code (new SqlConnection("...")). If any other connection detail changes, a developer would need to modify the source code, recompile the application, and redeploy it
   Performance: The query uses SELECT *, meaning it fetches all columns from the Customer table regardless of whether they are actually needed by the calling code. This can lead to performance inefficiencies, especially if the Customer table has many columns, large data types. Fetching unnecessary data increases network traffic, database load, and memory consumption on the application server

3. For each problem identified, briefly propose a specific improvement.
   Ans.
   Security: use Parameterized Query

From : var sql = "SELECT * FROM Customer WHERE id = '" + id + "'";
change to:
var sql = "SELECT * FROM Customer WHERE id = @id";
using (var cmd = new SqlCommand(sql, conn))
{
  cmd.Parameters.AddWithValue("@id", id);
  using (var da = new SqlDataAdapter(cmd))
  {
    da.Fill(dt);
  }
}
Maintainability : Use configuration
From : using (var conn = new SqlConnection("..."))
change to : string connectionString =
ConfigurationManager.ConnectionStrings["CustomerDBConnection"].ConnectionString;
using (var conn = new SqlConnection(connectionString)) { /* ... */ }
Performance: specifying the column names you need
From : SELECT *
change to : “SELECT id, name, email, transactions FROM Customer WHERE id = @id";

Part 4: System and User Perspective

What input fields would a user (e.g., a customer service representative) need on the

screen to search for customer data?

Ans. input field is customer id(Required) and date range (start date, end date) (when they want to know data in time)

After the search is performed, what would the output look like on their screen? (e.g., a

table, a profile card, a list of transactions).

Ans. Output would display a customer profile card and a list of transactions. Both of these would reflect any applied date range filters.