.NET Core console application

Here's a step-by-step guide to creating a .NET Core console application that uses SQLite for performing basic CRUD (Create, Read, Update, Delete) operations on an employee entity. This guide will also demonstrate how to improve the application by transitioning from a basic structure to a more organized OOP approach.

Here's a simpler version of the application that is more suited for beginners. This code performs CRUD operations directly using raw SQL queries without any Entity Framework, and everything is contained inside the `Main` method using a switch-case and an infinite loop for a menu-driven console application.

Steps :

- **Install .NET SDK**:
  - Make sure you have the .NET SDK installed. You can verify by running the following command:

    dotnet --version

- **Create a new Console App**:
  - Open a terminal/command prompt and create a new console application:

    dotnet new console -n EmployeeCRUDApp
    cd EmployeeCRUDApp

- **Install SQLite NuGet Package:**
  - Open your terminal or command prompt in the project directory and run the following command to add the SQLite NuGet package:

    dotnet add package Microsoft.Data.Sqlite

    Alternatively, if you're using Visual Studio, you can:

    - Right-click on your project in **Solution Explorer**.
    - Select **Manage NuGet Packages**.
    - Search for `Microsoft.Data.Sqlite` and install it.

    This will add the SQLite dependency to your `.csproj` file.

- **Steps to run app**
  - **dotnet restore**
  - **dotnet clean**
  - **dotnet build**
  - **dotnet run**

- Verify .csproj file

```
<Project Sdk="Microsoft.NET.Sdk">

  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.Data.Sqlite" Version="8.0.8" />
  </ItemGroup>

</Project>
```

- program.cs

```
using System;
using Microsoft.Data.Sqlite;

namespace EmployeeCRUDApp
{
    class Program
    {
        static void Main(string[] args)
        {
            bool exit = false;
            while (!exit)
            {
                Console.Clear();  // Clear the console before showing the menu
                Console.WriteLine("\n--- Employee Management ---");
                Console.WriteLine("1. Create Employee");
                Console.WriteLine("2. List Employees");
                Console.WriteLine("3. Update Employee");
                Console.WriteLine("4. Delete Employee");
                Console.WriteLine("5. Exit");
                Console.Write("Select an option: ");
                string option = Console.ReadLine();

                using (var connection = OpenConnection())
                {
                    switch (option)
                    {
                        case "1":
                            Console.Clear();  // Clear the screen before creating an
employee
                            CreateEmployee(connection);
                            break;
```

```
                    case "2":
                        Console.Clear();  // Clear the screen before listing employees
                        ListEmployees(connection);
                        break;
                    case "3":
                        Console.Clear();  // Clear the screen before updating an
employee
                        UpdateEmployee(connection);
                        break;
                    case "4":
                        Console.Clear();  // Clear the screen before deleting an
employee
                        DeleteEmployee(connection);
                        break;
                    case "5":
                        exit = true;
                        break;
                    default:
                        Console.WriteLine("Invalid option. Please try again.");
                        break;
                }

                if (!exit)
                {
                    Console.WriteLine("\nPress Enter to continue...");
                    Console.ReadLine();  // Pause for user to see the result before
clearing
                }
            }
        }
        // Open the SQLite connection
        static SqliteConnection OpenConnection()
        {
            var connection = new SqliteConnection("Data Source=employee.db;");
            connection.Open();

            string createTableQuery = @"CREATE TABLE IF NOT EXISTS Employees (
                                        EmployeeID INTEGER PRIMARY KEY AUTOINCREMENT,
                                        FirstName TEXT NOT NULL,
                                        LastName TEXT NOT NULL,
                                        DateOfBirth TEXT NOT NULL)";
            using (var command = new SqliteCommand(createTableQuery, connection))
            {
                command.ExecuteNonQuery();
            }

            return connection;
        }
```

```
        // Create a new employee
        static void CreateEmployee(SqliteConnection connection)
        {
            Console.Write("Enter First Name: ");
            string firstName = Console.ReadLine();

            Console.Write("Enter Last Name: ");
            string lastName = Console.ReadLine();

            Console.Write("Enter Date of Birth (yyyy-mm-dd): ");
            string dob = Console.ReadLine();

            string insertQuery = "INSERT INTO Employees (FirstName, LastName, DateOfBirth)
VALUES (@FirstName, @LastName, @DateOfBirth)";
            using (var command = new SqliteCommand(insertQuery, connection))
            {
                command.Parameters.AddWithValue("@FirstName", firstName);
                command.Parameters.AddWithValue("@LastName", lastName);
                command.Parameters.AddWithValue("@DateOfBirth", dob);
                command.ExecuteNonQuery();
            }

            Console.WriteLine("Employee created successfully.");
        }
        // List all employees
        static void ListEmployees(SqliteConnection connection)
        {
            string selectQuery = "SELECT * FROM Employees";
            using (var command = new SqliteCommand(selectQuery, connection))
            using (var reader = command.ExecuteReader())
            {
                Console.WriteLine("\n--- Employee List ---");
                while (reader.Read())
                {
                    Console.WriteLine($"ID: {reader["EmployeeID"]}, Name:
{reader["FirstName"]} {reader["LastName"]}, DOB: {reader["DateOfBirth"]}");
                }
            }
        }
        // Update an employee
        static void UpdateEmployee(SqliteConnection connection)
        {
            Console.Write("Enter Employee ID to update: ");
            int employeeID = int.Parse(Console.ReadLine());

            Console.Write("Enter new First Name: ");
            string newFirstName = Console.ReadLine();
```

```
            Console.Write("Enter new Last Name: ");
            string newLastName = Console.ReadLine();

            string updateQuery = "UPDATE Employees SET FirstName = @FirstName, LastName =
@LastName WHERE EmployeeID = @EmployeeID";
            using (var command = new SqliteCommand(updateQuery, connection))
            {
                command.Parameters.AddWithValue("@FirstName", newFirstName);
                command.Parameters.AddWithValue("@LastName", newLastName);
                command.Parameters.AddWithValue("@EmployeeID", employeeID);
                int rowsAffected = command.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine("Employee updated successfully.");
                }
                else
                {
                    Console.WriteLine("Employee not found.");
                }
            }
        }
        // Delete an employee
        static void DeleteEmployee(SqliteConnection connection)
        {
            Console.Write("Enter Employee ID to delete: ");
            int employeeID = int.Parse(Console.ReadLine());

            string deleteQuery = "DELETE FROM Employees WHERE EmployeeID = @EmployeeID";
            using (var command = new SqliteCommand(deleteQuery, connection))
            {
                command.Parameters.AddWithValue("@EmployeeID", employeeID);
                int rowsAffected = command.ExecuteNonQuery();
                if (rowsAffected > 0)
                {
                    Console.WriteLine("Employee deleted successfully.");
                }
                else
                {
                    Console.WriteLine("Employee not found.");
                }
            }
        }
    }
}
```