```
using Microsoft.EntityFrameworkCore; using Microsoft.Extensions.DependencyInjection;
using PracticeProblem_2.Data; var builder = WebApplication.CreateBuilder(args);
builder.Services.AddDbContext<OdersDbContext>(options => options.UseSqlServer(builder.Configuration.G
?? throw new InvalidOperationException(”Connection string ’OdersDbContext’
not found.”)));
```

```
// Add services to the container. builder.Services.AddControllersWithViews();
```

```
var app = builder.Build();
```

```
// Configure the HTTP request pipeline. if (!app.Environment.IsDevelopment())
{ app.UseExceptionHandler(”/Home/Error”); } app.UseStaticFiles();
```

```
app.UseRouting();
```

```
app.UseAuthorization();
```

```
app.MapControllerRoute( name: ”default”, pattern: ”{controller=Home}/{action=Index}/{id?}”);
```

```
app.Run();
```

```
{ ”Logging”: { ”LogLevel”: { ”Default”: ”Information”, ”Microsoft.AspNetCore”:
”Warning” } }, ”AllowedHosts”: ”*”, ”ConnectionStrings”: { ”OdersDbCon-
text”: ”server=DESKTOP-U064AL2;database=PloblemStatement_2;Trusted_Connection=True;MultipleAct
} }
```

```
<!DOCTYPE html> <html lang=”en”> <head> <meta charset=”utf-
8” /> <meta name=”viewport” content=”width=device-width, initial-
scale=1.0” /> <title>@ViewData[”Title”] - PracticeProblem_2</title>
<link rel=”stylesheet” href=”~/lib/bootstrap/dist/css/bootstrap.min.css” />
<link rel=”stylesheet” href=”~/css/site.css” asp-append-version=”true” />
<link rel=”stylesheet” href=”~/PracticeProblem_2.styles.css” asp-append-
version=”true” /> </head> <body> <header> <nav class=”navbar navbar-
expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-
shadow mb-3”> <div class=”container-fluid”> <a class=”navbar-brand” asp-
area=”” asp-controller=”Home” asp-action=”Index”>PracticeProblem_2</a>
<button class=”navbar-toggler” type=”button” data-bs-toggle=”collapse”
data-bs-target=”.navbar-collapse” aria-controls=”navbarSupportedContent”
aria-expanded=”false” aria-label=”Toggle navigation”> <span class=”navbar-
toggler-icon”></span> </button> <div class=”navbar-collapse collapse
d-sm-inline-flex justify-content-between”> <ul class=”navbar-nav flex-
grow-1”> <li class=”nav-item”> <a class=”nav-link text-dark” asp-area=””
asp-controller=”Home” asp-action=”Index”>Home</a> </li> <li class=”nav-
item”> <a class=”nav-link text-dark” asp-area=”” asp-controller=”Home”
asp-action=”Privacy”>Privacy</a> </li> <li class=”nav-item”> <a
class=”nav-link text-dark” asp-area=”” asp-controller=”Orders” asp-
action=”Index”>Orders</a> </li> </ul> </div> </div> </nav>
</header> <div class=”container”> <main role=”main” class=”pb-3”>
@RenderBody() </main> </div>
```

```
<footer class=”border-top footer text-muted”> <div class=”container”>
&copy; 2024 - PracticeProblem_2 - <a asp-area=”” asp-controller=”Home”
asp-action=”Privacy”>Privacy</a> </div> </footer> <script src=”~/lib/jquery/dist/jquery.min.js”></scr
<script src=”~/lib/bootstrap/dist/js/bootstrap.bundle.min.js”></script>
<script src=”~/js/site.js” asp-append-version=”true”></script> @await
RenderSectionAsync(”Scripts”, required: false) </body> </html>

@{ ViewData[”Title”] = ”Home Page”; }

<div class=”text-center”> <h1 class=”display-4”>Welcome to Joe’s
Pizza!</h1> <p>Learn about <a href=”https://docs.microsoft.com/aspnet/core”>building
Web apps with ASP.NET Core</a>.</p> </div>

using System.ComponentModel.DataAnnotations; using System.ComponentModel.DataAnnotations.Schema;

namespace PracticeProblem_2.Models { [Table(”Orders”)] public class Orders
{ [Key] public int OId { get; set; } public string OName { get; set; } public
string COrder { get; set; } public string OAddress { get; set; }

} }

using System; using System.Collections.Generic; using System.Linq; us-
ing System.Threading.Tasks; using Microsoft.AspNetCore.Mvc; using Mi-
crosoft.AspNetCore.Mvc.Rendering; using Microsoft.EntityFrameworkCore;
using PracticeProblem_2.Data; using PracticeProblem_2.Models;

namespace PracticeProblem_2.Controllers { public class OrdersController :
Controller { private readonly OdersDbContext _context;

public OrdersController(OdersDbContext context) { _context = context; }

// GET: Orders public async Task<IActionResult> Index() { return
_context.Orders != null ? View(await _context.Orders.ToListAsync()) :
Problem(”Entity set ’OdersDbContext.Orders’ is null.”); }

// GET: Orders/Details/5 public async Task<IActionResult> Details(int? id)
{ if (id == null || _context.Orders == null) { return NotFound(); }

var orders = await _context.Orders .FirstOrDefaultAsync(m => m.OId ==
id); if (orders == null) { return NotFound(); }

return View(orders); }

// GET: Orders/Create public IActionResult Create() { return View(); }

// POST: Orders/Create // To protect from overposting attacks, en-
able the specific properties you want to bind to. // For more de-
tails, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost]
[ValidateAntiForgeryToken] public async Task<IActionResult> Cre-
ate([Bind(”OId,Name,Order,Address”)] Orders orders) { if (ModelState.IsValid)
{ _context.Add(orders); await _context.SaveChangesAsync(); return Redirect-
ToAction(nameof(Index)); } return View(orders); }
```

// GET: Orders/Edit/5 public async Task<IActionResult> Edit(int? id) { if (id == null || _context.Orders == null) { return NotFound(); }

var orders = await _context.Orders.FindAsync(id); if (orders == null) { return NotFound(); } return View(orders); }

// POST: Orders/Edit/5 // To protect from overposting attacks, enable the specific properties you want to bind to. // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598. [HttpPost] [ValidateAntiForgeryToken] public async Task<IActionResult> Edit(int id, [Bind(”OId,Name,Order,Address”)] Orders orders) { if (id != orders.OId) { return NotFound(); }

if (ModelState.IsValid) { try { _context.Update(orders); await _context.SaveChangesAsync(); } catch (DbUpdateConcurrencyException) { if (!OrdersExists(orders.OId)) { return NotFound(); } else { throw; } } return RedirectToAction(nameof(Index)); } return View(orders); }

// GET: Orders/Delete/5 public async Task<IActionResult> Delete(int? id) { if (id == null || _context.Orders == null) { return NotFound(); }

var orders = await _context.Orders .FirstOrDefaultAsync(m => m.OId == id); if (orders == null) { return NotFound(); }

return View(orders); }

// POST: Orders/Delete/5 [HttpPost, ActionName(”Delete”)] [ValidateAntiForgeryToken] public async Task<IActionResult> DeleteConfirmed(int id) { if (_context.Orders == null) { return Problem(”Entity set ’OdersDbContext.Orders’ is null.”); } var orders = await _context.Orders.FindAsync(id); if (orders != null) { _context.Orders.Remove(orders); }

await _context.SaveChangesAsync(); return RedirectToAction(nameof(Index)); }

private bool OrdersExists(int id) { return (_context.Orders?.Any(e => e.OId == id)).GetValueOrDefault(); } } }