# EF Core 8.0 Hands-On Labs – Week 3 (HOL-Aligned)

Name: Pranjal Yadav(22052918), SupersetID: 6363586  
Program: Cognizant Digital Nurture 4.0 – DotNet FSE Deep Skilling  
Project: Retail Inventory System  
Technology: Entity Framework Core 8.0 (SQLite)

----------------------------------------------------------------------------------------------------------------------

## Lab 1: Understanding ORM with EF Core

Objective:  
Understand how EF Core maps C# model classes to relational database tables using DbContext.

🧾 Code

Models/Product.cs

namespace RetailInventorySystem.Models;  
  
public class Product  
{  
 public int Id { get; set; }  
 public string Name { get; set; } = string.Empty;  
 public decimal Price { get; set; }  
 public int CategoryId { get; set; }  
 public Category? Category { get; set; }  
}

Models/Category.cs

namespace RetailInventorySystem.Models;  
  
public class Category  
{  
 public int Id { get; set; }  
 public string Name { get; set; } = string.Empty;  
 public ICollection<Product>? Products { get; set; }  
}

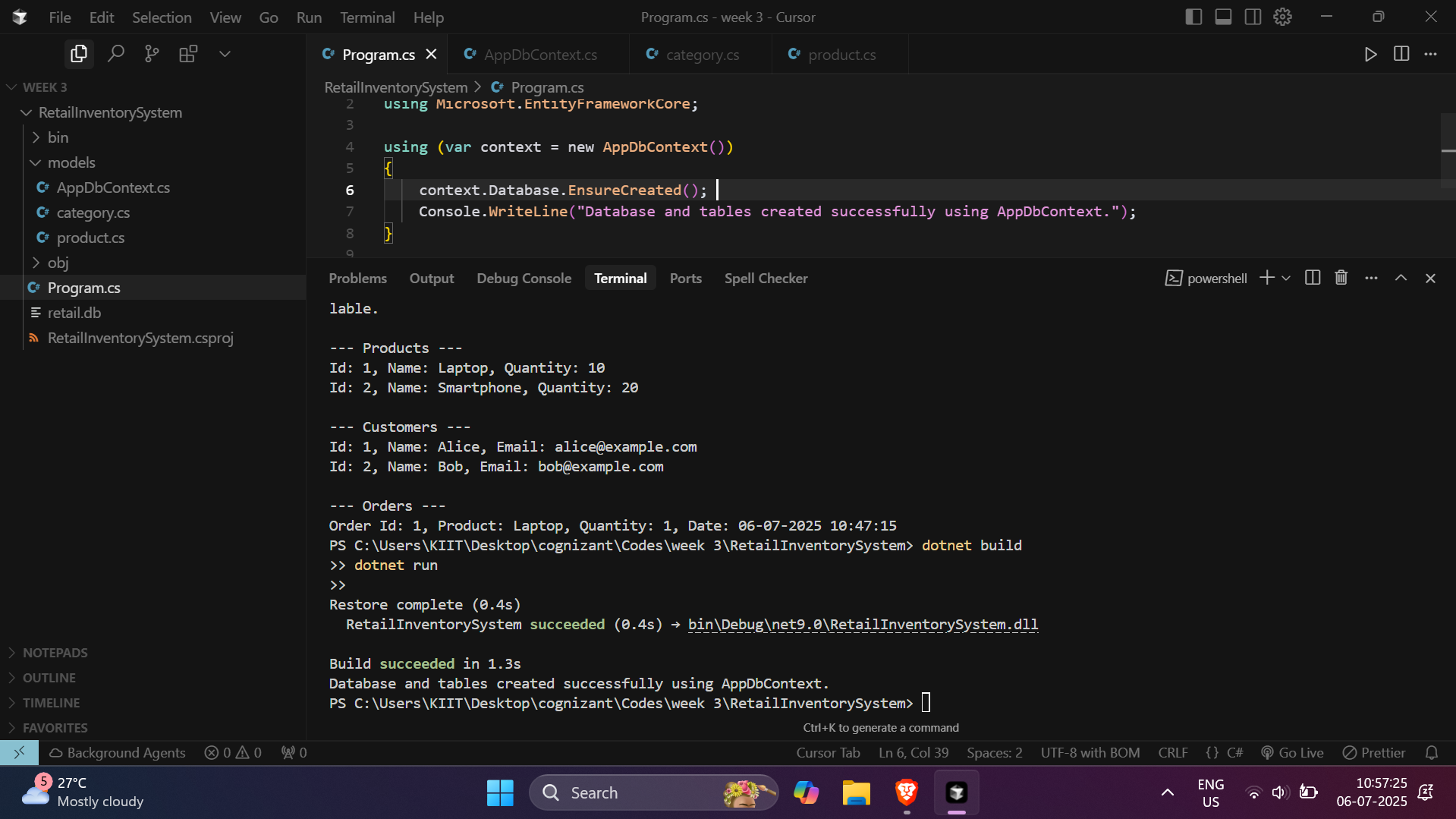
Models/AppDbContext.cs

using Microsoft.EntityFrameworkCore;  
  
namespace RetailInventorySystem.Models;  
  
public class AppDbContext : DbContext  
{  
 public DbSet<Product> Products => Set<Product>();  
 public DbSet<Category> Categories => Set<Category>();  
  
 protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  
 {  
 optionsBuilder.UseSqlite("Data Source=retail.db");  
 }  
}

Program.cs

using RetailInventorySystem.Models;  
using Microsoft.EntityFrameworkCore;  
  
using (var context = new AppDbContext())  
{  
 context.Database.EnsureCreated();  
 Console.WriteLine("Database and tables created successfully using AppDbContext.");  
}

Screenshot Placeholder: Lab 1 – Database Created

---

## Lab 2: Creating Models and Applying Migrations

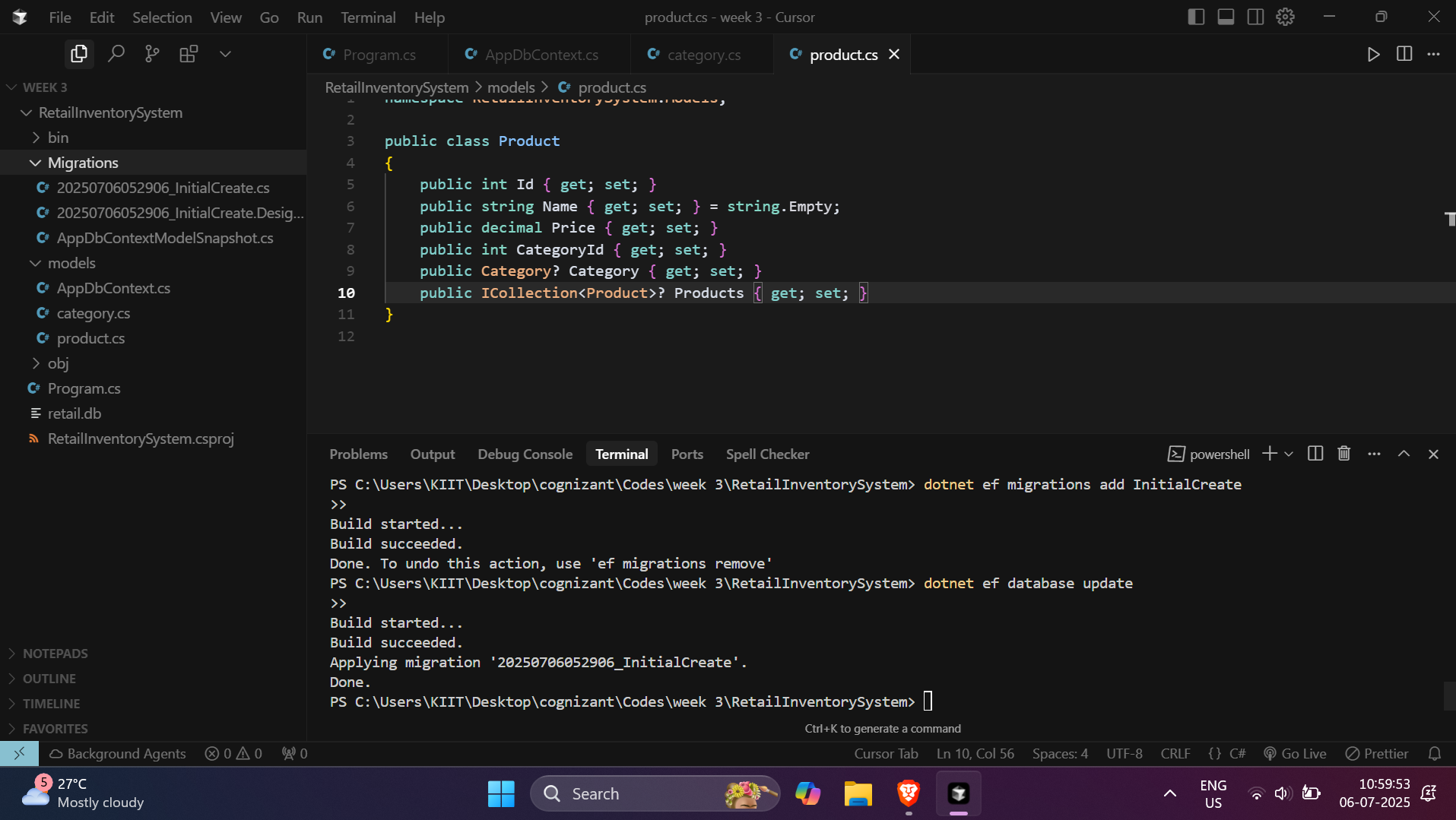
Objective:  
Create model relationships and generate schema using EF Core migrations.

🧾 Code

CLI Commands

dotnet ef migrations add InitialCreate  
dotnet ef database update

Screenshot Placeholder: Lab 2 – Migration Applied Successfully



## Lab 3: Inserting Data using Async Methods

Objective:  
Use AddAsync() and SaveChangesAsync() to insert sample Category and Product records.

🧾 Code

Program.cs

using RetailInventorySystem.Models;  
using Microsoft.EntityFrameworkCore;  
  
using (var context = new AppDbContext())  
{  
 await context.Database.EnsureCreatedAsync();  
  
 if (!await context.Categories.AnyAsync())  
 {  
 var electronics = new Category { Name = "Electronics" };  
 var groceries = new Category { Name = "Groceries" };  
  
 await context.Categories.AddRangeAsync(electronics, groceries);  
 await context.SaveChangesAsync();  
  
 var laptop = new Product { Name = "Laptop", Price = 50000, CategoryId = electronics.Id };  
 var rice = new Product { Name = "Rice", Price = 1500, CategoryId = groceries.Id };  
  
 await context.Products.AddRangeAsync(laptop, rice);  
 await context.SaveChangesAsync();  
  
 Console.WriteLine(" Sample data inserted successfully.");  
 }  
 else  
 {  
 Console.WriteLine("Data already exists.");  
 }  
}

📸 Screenshot Placeholder: Lab 3 – Sample Data Inserted

---

## Lab 4: Retrieving and Displaying Data

Objective:  
Use Include(), ToListAsync(), and LINQ queries to retrieve and display data, including related entities.

🧾 Code

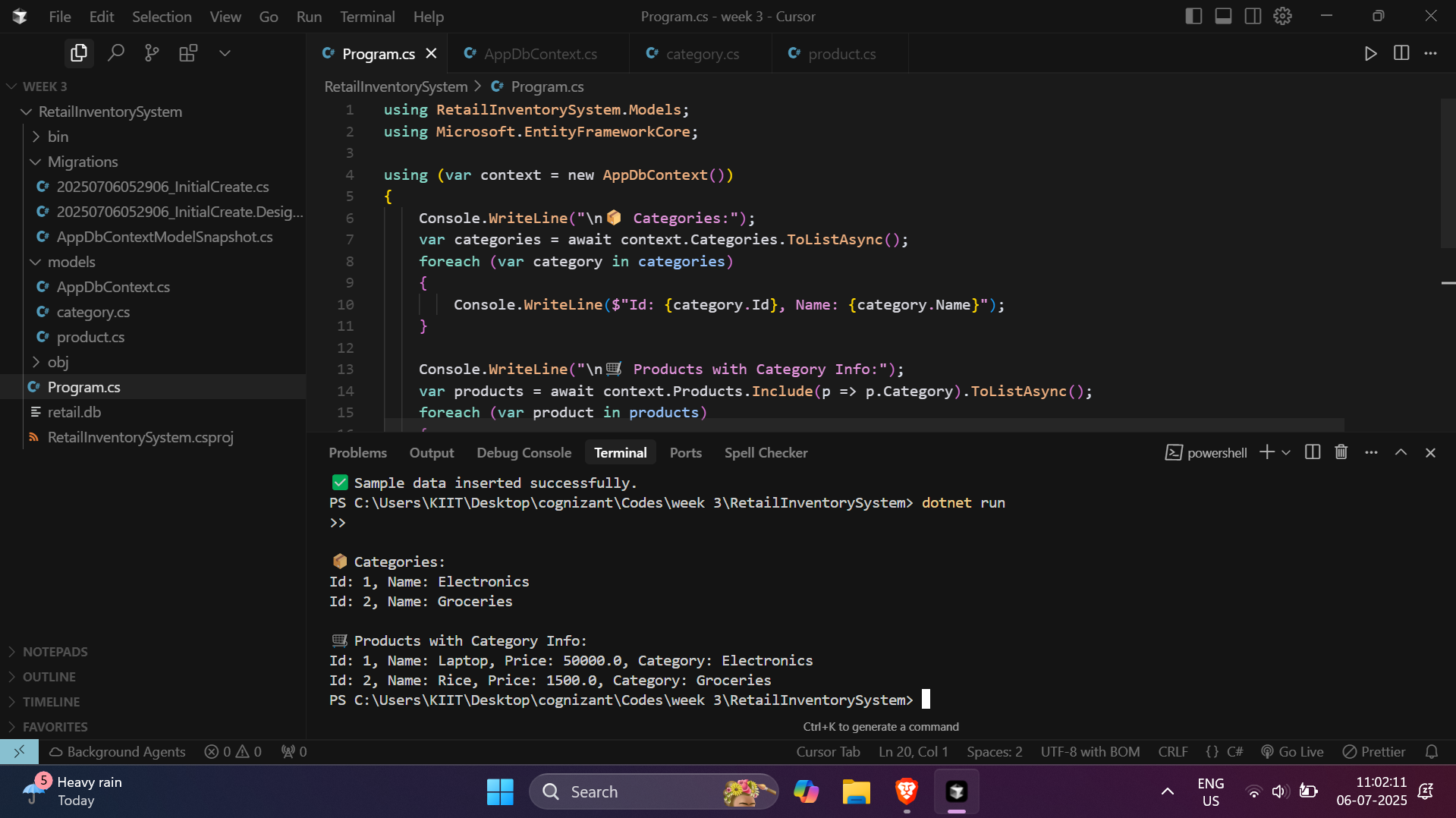
Program.cs

using RetailInventorySystem.Models;  
using Microsoft.EntityFrameworkCore;  
  
using (var context = new AppDbContext())  
{  
 var productsWithCategory = await context.Products  
 .Include(p => p.Category)  
 .ToListAsync();  
  
 foreach (var product in productsWithCategory)  
 {  
 Console.WriteLine($"Product: {product.Name}, Price: ₹{product.Price}, Category: {product.Category?.Name}");  
 }  
}

Expected Output

Product: Laptop, Price: ₹50000, Category: Electronics  
Product: Rice, Price: ₹1500, Category: Groceries

📸 Screenshot Placeholder: Lab 4 – Data Retrieved with Category



## Lab 5: Updating and Deleting Records

Objective:  
Update and delete records using Update(), Remove(), and SaveChangesAsync() methods.

🧾 Code

Program.cs

using RetailInventorySystem.Models;  
using Microsoft.EntityFrameworkCore;  
  
using (var context = new AppDbContext())  
{  
   
 var productToUpdate = await context.Products.FirstOrDefaultAsync(p => p.Name == "Laptop");  
 if (productToUpdate != null)  
 {  
 productToUpdate.Price = 48000;  
 context.Products.Update(productToUpdate);  
 await context.SaveChangesAsync();  
 Console.WriteLine(" Product price updated successfully.");  
 }  
  
 // Delete Example  
 var productToDelete = await context.Products.FirstOrDefaultAsync(p => p.Name == "Rice");  
 if (productToDelete != null)  
 {  
 context.Products.Remove(productToDelete);  
 await context.SaveChangesAsync();  
 Console.WriteLine("🗑️ Product deleted successfully.");  
 }  
}

Expected Output

✅ Product price updated successfully.  
🗑️ Product deleted successfully.

📸 Screenshot Placeholder: Lab 5 – Update & Delete Confirmation

