**Entity Framework Core 8.0**

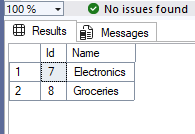
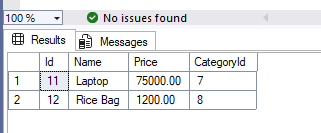
**Create Models**

| **using System.ComponentModel.DataAnnotations;  namespace RetailInventory.Models;  public class Category {  public int Id { get; set; }    [Required]  public string Name { get; set; } = string.Empty;    public ICollection<Product> Products { get; set; } = new List<Product>(); }** |
| --- |

| **using System.ComponentModel.DataAnnotations;  namespace RetailInventory.Models;  public class Product {  public int Id { get; set; }    [Required]  public string Name { get; set; } = string.Empty;    public decimal Price { get; set; }    public int CategoryId { get; set; }    public Category Category { get; set; } = null!; }** |
| --- |

**Create AppDbContext**

| **using Microsoft.EntityFrameworkCore; using RetailInventory.Models;  namespace RetailInventory.Data;  public class AppDbContext : DbContext {  public AppDbContext(DbContextOptions<AppDbContext> options) : base(options)  {  }   public DbSet<Product> Products { get; set; }  public DbSet<Category> Categories { get; set; }    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  if (!optionsBuilder.IsConfigured)  {  optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailInventoryDb;Trusted\_Connection=true;TrustServerCertificate=true;");  }  } }** |
| --- |

**Verify in SQL Server (SSMS):**

**Insert Data in** [**Program.cs**](http://program.cs)**:**

| **using Microsoft.EntityFrameworkCore; using RetailInventory.Data; using RetailInventory.Models;  namespace RetailInventory;  class Program {  static async Task Main(string[] args)  {  // Create DbContext with options  var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();  optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailInventoryDb;Trusted\_Connection=true;TrustServerCertificate=true;");    using var context = new AppDbContext(optionsBuilder.Options);   // Ensure database exists and has data  await context.Database.EnsureCreatedAsync();    // Add sample data if not exists  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 product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };  var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };  await context.Products.AddRangeAsync(product1, product2);  await context.SaveChangesAsync();  }**  **}**  **}** |
| --- |

**Retrieving Data from the Database**

**(Retrieve All Products,Find by ID,FirstOrDefault with Condition)**

| **using Microsoft.EntityFrameworkCore; using RetailInventory.Data; using RetailInventory.Models;  namespace RetailInventory;  class Program {  static async Task Main(string[] args)  {  // Create DbContext with options  var optionsBuilder = new DbContextOptionsBuilder<AppDbContext>();  optionsBuilder.UseSqlServer("Server=(localdb)\\mssqllocaldb;Database=RetailInventoryDb;Trusted\_Connection=true;TrustServerCertificate=true;");    using var context = new AppDbContext(optionsBuilder.Options);   // Ensure database exists and has data  await context.Database.EnsureCreatedAsync();    // Add sample data if not exists  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 product1 = new Product { Name = "Laptop", Price = 75000, Category = electronics };  var product2 = new Product { Name = "Rice Bag", Price = 1200, Category = groceries };  await context.Products.AddRangeAsync(product1, product2);  await context.SaveChangesAsync();  }   // Lab 5: Retrieving Data from the Database    // Step 1: Retrieve All Products  var products = await context.Products.ToListAsync();  foreach (var p in products)  Console.WriteLine($"{p.Name} - ₹{p.Price}");   // Step 2: Find by ID (use first product's ID)  var firstProduct = await context.Products.FirstOrDefaultAsync();  var product = await context.Products.FindAsync(firstProduct?.Id);  Console.WriteLine($"Found: {product?.Name ?? "Product not found"}");   // Step 3: FirstOrDefault with Condition  var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);  Console.WriteLine($"Expensive: {expensive?.Name ?? "No expensive product found"}");  } }** |
| --- |

**Output**

| **Laptop - ?75000.00 Rice Bag - ?1200.00 Found: Laptop Expensive: Laptop  C:\Users\bhowm\source\repos\AspireApp1\RetailInventory\bin\Debug\net9.0\RetailInventory.exe (process 22988) exited with code 0 (0x0). To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops. Press any key to close this window . . .** |
| --- |