LAB 3 : EF CORE 8.0 HOL

**Aim:** Using EF Core CLI to Create and Apply Migrations

**Scenario:** The retail store's database needs to be created based on the models you've defined.

You’ll use EF Core CLI to generate and apply migrations.

**Models\Product.cs:**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace RetailInventoryApp.Models

{

public class Product

{

public int ProductId { get; set; }

public string ProductName { get; set; } = string.Empty;

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; } = null!;

public int SupplierId { get; set; }

public Supplier Supplier { get; set; } = null!;

public Stock Stock { get; set; } = null!; navigation property

}

}

**Models\Category.cs:**

namespace RetailInventoryApp.Models

{

public class Category

{

public int CategoryId { get; set; }

public required string Name { get; set; } = string.Empty;

public ICollection<Product> Products { get; set; } = new List<Product>();

}

}

**Models\Stock.cs:**

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace RetailInventoryApp.Models

{

public class Stock

{

public int StockId { get; set; }

public int ProductId { get; set; }

public Product Product { get; set; } = null!;

public int QuantityAvailable { get; set; }

public DateTime LastChecked { get; set; }

}

}

**Models\Supplier.cs:**

namespace RetailInventoryApp.Models

{

public class Supplier

{

public int SupplierId { get; set; }

public required string SupplierName { get; set; } = string.Empty;

public required string ContactEmail { get; set; } = string.Empty;

public ICollection<Product> Products { get; set; } = new List<Product>();

}

}

**Data\RetailContext.cs:**

using Microsoft.EntityFrameworkCore;

using RetailInventoryApp.Models;

namespace RetailInventoryApp.Data

{

public class RetailContext : DbContext

{

public DbSet<Product> Products { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Supplier> Suppliers { get; set; }

public DbSet<Stock> Stocks { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlite("Data Source=RetailStore.db");

}

}

}

**Program.cs:**

using RetailInventoryApp.Data;

using RetailInventoryApp.Models;

using Microsoft.Data.Sqlite;

class Program

{

static void Main(string[] args)

{

using (var context = new RetailContext())

{

context.Database.EnsureCreated();

Console.WriteLine("Database setup complete with sample data.\n");

if (!context.Categories.Any())

{

var category1 = new Category

{

Name = "Beverages",

Products = new List<Product>()

};

var category2 = new Category

{

Name = "Groceries",

Products = new List<Product>()

};

var supplier1 = new Supplier

{

SupplierName = "Global Beverages",

ContactEmail = "contact@globalbev.com",

Products = new List<Product>()

};

var supplier2 = new Supplier

{

SupplierName = "Fresh Farm Foods",

ContactEmail = "info@freshfarm.com",

Products = new List<Product>()

};

var product1 = new Product

{

ProductName = "Orange Juice",

Price = 49.99M,

Category = category1,

Supplier = supplier1

};

var product2 = new Product

{

ProductName = "Whole Wheat Rice",

Price = 72.50M,

Category = category2,

Supplier = supplier2

};

var stock1 = new Stock

{

QuantityAvailable = 100,

LastChecked = DateTime.Now,

Product = product1

};

product1.Stock = stock1;

var stock2 = new Stock

{

QuantityAvailable = 60,

LastChecked = DateTime.Now,

Product = product2

};

product2.Stock = stock2;

category1.Products.Add(product1);

category2.Products.Add(product2);

supplier1.Products.Add(product1);

supplier2.Products.Add(product2);

context.Categories.AddRange(category1, category2);

context.Suppliers.AddRange(supplier1, supplier2);

context.Products.AddRange(product1, product2);

context.Stocks.AddRange(stock1, stock2);

context.SaveChanges();

}

Console.WriteLine("=== CATEGORIES ===");

foreach (var category in context.Categories)

{

Console.WriteLine($"ID: {category.CategoryId}, Name: {category.Name}");

}

Console.WriteLine("\n=== SUPPLIERS ===");

foreach (var supplier in context.Suppliers)

{

Console.WriteLine($"ID: {supplier.SupplierId}, Name: {supplier.SupplierName}, Email: {supplier.ContactEmail}");

}

Console.WriteLine("\n=== PRODUCTS ===");

foreach (var product in context.Products)

{

Console.WriteLine($"ID: {product.ProductId}, Name: {product.ProductName}, Price: {product.Price}, " +

$"Category: {product.Category.Name}, Supplier: {product.Supplier.SupplierName}");

}

Console.WriteLine("\n=== STOCKS ===");

foreach (var stock in context.Stocks)

{

Console.WriteLine($"Stock ID: {stock.StockId}, Quantity: {stock.QuantityAvailable}, " +

$"Last Checked: {stock.LastChecked}, Product: {stock.Product.ProductName}");

}

}

string dbPath = "RetailStoreLite.db";

string connectionString = $"Data Source={dbPath}";

using var connection = new SqliteConnection(connectionString);

connection.Open();

Console.WriteLine("\nConnected to SQLite database via raw query.\n");

var getTablesCommand = connection.CreateCommand();

getTablesCommand.CommandText = "SELECT name FROM sqlite\_master WHERE type='table' AND name NOT LIKE 'sqlite\_%';";

using var tablesReader = getTablesCommand.ExecuteReader();

while (tablesReader.Read())

{

string tableName = tablesReader.GetString(0);

Console.WriteLine($"\n📋 TABLE: {tableName}");

var getContentCommand = connection.CreateCommand();

getContentCommand.CommandText = $"SELECT \* FROM {tableName};";

using var contentReader = getContentCommand.ExecuteReader();

for (int i = 0; i < contentReader.FieldCount; i++)

{

Console.Write($"{contentReader.GetName(i),-20}");

}

Console.WriteLine();

while (contentReader.Read())

{

for (int i = 0; i < contentReader.FieldCount; i++)

{

Console.Write($"{contentReader.GetValue(i),-20}");

}

Console.WriteLine();

}

}

connection.Close();

}

}

**Output:**

