**Cognizant - DN 4.0 I Deep Skilling**

**WEEK-3**

**Entity Framework Core- 8.0**

### ****Lab 1 – Understanding ORM with a Retail Inventory System****

**Scenario:**  
You are developing an inventory management system for a retail store. The store wants to track **products**, **categories**, and **stock levels** in a SQL Server database.

**Question:** Explain what Object-Relational Mapping (ORM) is and how EF Core 8.0 helps bridge the gap between C# objects and relational tables.Also, compare EF Core with EF Framework (EF6), and list three key features introduced in EF Core 8.0.

### ****Lab 2 – Setting Up the Database Context for a Retail Store****

**Scenario:**  
The retail store wants to store product and category data in a SQL Server database.

**Question:** Define the C# models for Product and Category, then create a DbContext class named AppDbContext to represent the database.  
Configure it to connect to a SQL Server instance using EF Core and specify the necessary DbSet properties.

### ****Lab 3 – Using EF Core CLI to Create and Apply Migrations****

**Scenario:**  
You’ve defined models for Product and Category. Now the store wants the actual database to be created based on those models.

**Question:**

Using the EF Core CLI, how would you:

1. Create the first migration for the schema?
2. Apply the migration to create the SQL Server database and tables?
3. Also, mention how you would verify in SQL Server that the Products and Categories tables were successfully created.

### ****Lab 4 – Inserting Initial Data into the Database****

**Scenario:**  
The store manager wants to add initial product categories and products such as “Electronics”, “Groceries”, “Laptop”, and “Rice Bag”.

**Question:** Using EF Core, how would you insert initial data into the Categories and Products tables in an asynchronous way using AddRangeAsync() and SaveChangesAsync()?

Write sample code for inserting this data in the Program.cs file of your console application.

### ****Lab 5 – Retrieving Data from the Database****

**Scenario:**  
The store wants to display product details on the dashboard, including all product listings, searching by product ID, and identifying expensive products.

**Question:**

Using EF Core, how would you:

1. Retrieve all products from the database and display their names and prices?
2. Find a product by its ID?
3. Find the first product whose price is greater than ₹50,000?
4. Use asynchronous methods such as ToListAsync(), FindAsync(), and FirstOrDefaultAsync() to perform these queries.

**SOLUTION :**

**CODE -:**

**Class Product.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailsInventory

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int CategoryId { get; set; }

public Category Category { get; set; }

}

}

**Class Category.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RetailsInventory

{

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

}

}

**Class Program.cs:**

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using RetailsInventory;

using System;

using System.Threading.Tasks;

class Program

{

static async Task Main(string[] args)

{

var host = Host.CreateDefaultBuilder(args)

.ConfigureAppConfiguration(config =>

{

config.AddJsonFile("appsettings.json", optional: false);

})

.ConfigureServices((context, services) =>

{

services.AddDbContext<AppDbContext>();

})

.Build();

using var scope = host.Services.CreateScope();

var context = scope.ServiceProvider.GetRequiredService<AppDbContext>();

// Insert initial data (Lab 4)

var electronics = new Category { Name = "Electronics" };

var groceries = new Category { Name = "Groceries" };

await context.Categories.AddRangeAsync(electronics, groceries);

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();

Console.WriteLine("Inserted initial data.");

// Retrieve data (Lab 5)

Console.WriteLine("\nAll Products:");

var products = await context.Products.ToListAsync();

foreach (var p in products)

Console.WriteLine($"{p.Name} - ₹{p.Price}");

Console.WriteLine("\nFind Product by ID:");

var product = await context.Products.FindAsync(1);

Console.WriteLine($"Found: {product?.Name}");

Console.WriteLine("\nFirst Expensive Product:");

var expensive = await context.Products.FirstOrDefaultAsync(p => p.Price > 50000);

Console.WriteLine($"Expensive: {expensive?.Name}");

}

}

**AppContextDb.cs**

using Microsoft.EntityFrameworkCore;

namespace RetailsInventory

{

public class AppDbContext : DbContext

{

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("Server=BT-2205664\\SQLEXPRESS;Database=RetailDbEFCore;Trusted\_Connection=True;");

}

}

}

}

**JSON File:**

{

"ConnectionStrings": {

"DefaultConnection": "Server=BT-2205664\\SQLEXPRESS;Database=RetailDbEFCore;Trusted\_Connection=True;"

}

}

**OUTPUT -:**

