**Lab 1: Understanding ORM with a Retail Inventory System**

**What is ORM?**

**ORM (Object-Relational Mapping)** is a technique that allows developers to work with databases using object-oriented programming principles. In C#, an ORM like **Entity Framework Core (EF Core)** maps C# classes to database tables.

**How ORM Maps C# Classes to Tables**

**Class = Table**  
Each C# class represents a table in the database.

**Property = Column**  
Each property in the class becomes a column.

**Object = Row**  
An instance of the class represents a row in the table.

public class Product

{

public int Id { get; set; } // Primary Key

public string Name { get; set; }

public int StockLevel { get; set; }

}

CREATE TABLE Products (

Id INT PRIMARY KEY,

Name NVARCHAR(MAX),

StockLevel INT

);

| **Feature** | **EF Core** | **EF Framework (EF6)** |
| --- | --- | --- |
| Platform | Cross-platform (.NET 6/7/8) | Windows only (.NET Framework) |
| Lightweight | Yes | No |
| LINQ, Async, Compiled Queries | Fully supported | Limited support |
| Performance | Better with compiled models | Slower |
| JSON Support | Yes (EF Core 8) | No |

**Create a new Console App**

dotnet new console -n RetailInventory

cd RetailInventory

dotnet build

dotnet run

dotnet run

**Install EF Core Packages**

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Design

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

public class Product

{

public int Id { get; set; } // Primary key

public string Name { get; set; }

public int Quantity { get; set; }

public decimal Price { get; set; }

}

// RetailContext.cs

using Microsoft.EntityFrameworkCore;

public class RetailContext : DbContext

{

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

optionsBuilder.UseSqlServer(@"Server=(localdb)\MSSQLLocalDB;Database=RetailStoreDB;Trusted\_Connection=True;");

}

}

using System;

using System.Linq;

class Program

{

static void Main()

{

using (var context = new RetailContext())

{

if (!context.Products.Any())

{

context.Products.Add(new Product { Name = "Laptop", Quantity = 10, Price = 75000 });

context.Products.Add(new Product { Name = "Mouse", Quantity = 50, Price = 500 });

context.SaveChanges();

}

// Display all products

var products = context.Products.ToList();

Console.WriteLine("Product List:");

foreach (var p in products)

{

Console.WriteLine($"ID: {p.Id}, Name: {p.Name}, Quantity: {p.Quantity}, Price: ₹{p.Price}");

}

}

}

}

**Create the Initial migration**

dotnet ef migrations add InitialCreate

dotnet ef database update

**OUTPUT**

using System;

using System.Linq;

class Program

{

static void Main()

{

using var context = new RetailContext();

// Seed a product

if (!context.Products.Any())

{

context.Products.Add(new Product { Name = "Headphones", Quantity = 15, Price = 2500 });

context.SaveChanges();

}

// Display products

Console.WriteLine("Product List:");

foreach (var p in context.Products.ToList())

{

Console.WriteLine($"{p.Id}: {p.Name} - Qty: {p.Quantity}, Price: ₹{p.Price}");

}

}

}

**Lab 4: Inserting Initial Data into the Database**

// Program.cs

using System;

using System.Linq;

class Program

{

static void Main()

{

using var context = new RetailContext();

// Insert initial data only if the table is empty

if (!context.Products.Any())

{

var products = new[]

{

new Product { Name = "Laptop", Quantity = 10, Price = 75000 },

new Product { Name = "Mouse", Quantity = 50, Price = 500 },

new Product { Name = "Keyboard", Quantity = 30, Price = 1500 },

new Product { Name = "Monitor", Quantity = 20, Price = 12000 }

};

context.Products.AddRange(products);

context.SaveChanges();

Console.WriteLine("Initial data inserted successfully.");

}

else

{

Console.WriteLine("Database already contains data. Skipping insertion.");

}

// Display the data

Console.WriteLine("\nCurrent Products in Inventory:");

foreach (var product in context.Products.ToList())

{

Console.WriteLine($"{product.Id}: {product.Name} - Qty: {product.Quantity}, Price: ₹{product.Price}");

}

}

}

Output

Initial data inserted successfully.

Current Products in Inventory:

1: Laptop - Qty: 10, Price: ₹75000

2: Mouse - Qty: 50, Price: ₹500

3: Keyboard - Qty: 30, Price: ₹1500

4: Monitor - Qty: 20, Price: ₹12000

**Lab 5: Retrieving Data from the Database**

using System;

using System.Linq;

class Program

{

static void Main()

{

using var context = new RetailContext();

// Get all products

var products = context.Products.ToList();

Console.WriteLine("📦 Product Inventory:");

foreach (var p in products)

{

Console.WriteLine($"ID: {p.Id}, Name: {p.Name}, Quantity: {p.Quantity}, Price: ₹{p.Price}");

}

}

}

OUTPUT

Product Inventory:

ID: 1, Name: Laptop, Quantity: 10, Price: ₹75000

ID: 2, Name: Mouse, Quantity: 50, Price: ₹500

ID: 3, Name: Keyboard, Quantity: 30, Price: ₹1500