**Lab 3: 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.**

**Objective: Learn how to use EF Core CLI to manage database schema changes.**

**Creating Models**

**Category.cs**

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

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

}

**Product.cs**

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; }

}

**Configuring DbContext**

using Microsoft.EntityFrameworkCore;

public class AppDbContext : DbContext {

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

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {

optionsBuilder.UseSqlServer("Your\_Connection\_String\_Here");

}

}

**Using EF Core CLI**

Objective:  
Use EF Core CLI to create and apply database migrations.

Steps:

1. Open Command Prompt (Windows + R → type cmd)
2. Navigate to your project folder:

cd C:\Users\welcome\Source\Repos\RetailInventory\RetailInventory

1. Install EF CLI globally (only once):

dotnet tool install --global dotnet-ef

1. Create migration:

dotnet ef migrations add InitialCreate

1. Apply migration to create the database:

dotnet ef database update

**Verifying in SSMS**

Steps:

1. Open SQL Server Management Studio (SSMS)
2. Connect to: (localdb)\MSSQLLocalDB
3. Expand Databases → RetailInventory
4. Expand Tables → You will see:
   * Categories
   * Products
5. Right-click each table → Select "Select Top 1000 Rows" to view sample data.

**Verifying Output in Visual Studio**

Steps:

1. Press Ctrl + F5 in Visual Studio to run the program

**OUTPUT :**