**Step 1: Create Database and Tables**

1. Open terminal and connect to SQL Server:

sqlcmd -U sa -P examlyMssql@123

1. Create the database:

CREATE DATABASE appdb;

GO

USE appdb;

GO

1. Create **Dept** table:

CREATE TABLE Dept (

Id INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(30) NOT NULL,

Location VARCHAR(25) NOT NULL

);

GO

1. Create **Employee** table with foreign key:

CREATE TABLE Employee (

Id INT IDENTITY(1,1) PRIMARY KEY,

Name VARCHAR(25) NOT NULL,

Email VARCHAR(40) NOT NULL,

DeptId INT NOT NULL,

DateOfBirth DATE NOT NULL,

Salary INT NOT NULL,

CONSTRAINT FK\_Employee\_Dept FOREIGN KEY (DeptId) REFERENCES Dept(Id)

);

GO

1. Insert sample data:

INSERT INTO Dept (Name, Location) VALUES

('IT', 'New York'),

('HR', 'Los Angeles'),

('Finance', 'Chicago');

GO

INSERT INTO Employee (Name, Email, DeptId, DateOfBirth, Salary) VALUES

('John Doe', 'john@example.com', 1, '1990-05-12', 50000),

('Jane Smith', 'jane@example.com', 2, '1988-03-22', 45000),

('Mark Brown', 'mark@example.com', 1, '1992-07-15', 55000);

GO

**Step 2: Setup .NET Project**

1. Create a new Web API project:

dotnet new webapi -n EmployeeDeptApp

cd EmployeeDeptApp

1. Install EF Core tools (if not already):

dotnet new tool-manifest

dotnet tool install --local dotnet-ef --version 6.0.6

dotnet dotnet-ef

**Step 3: Generate Models using DB-First Approach**

Run the **scaffold command**:

dotnet dotnet-ef dbcontext scaffold "User ID=sa;password=examlyMssql@123;Server=localhost;Database=appdb;Trusted\_Connection=False;Persist Security Info=False;Encrypt=False" Microsoft.EntityFrameworkCore.SqlServer -o Models

This will generate:

* Dept.cs → Dept model
* Employee.cs → Employee model
* appdbContext.cs → DbContext class

**Step 4: Verify Models and Relationships**

**Dept.cs**

public partial class Dept

{

public Dept()

{

Employees = new HashSet<Employee>();

}

public int Id { get; set; }

public string Name { get; set; } = null!;

public string Location { get; set; } = null!;

public virtual ICollection<Employee> Employees { get; set; }

}

**Employee.cs**

public partial class Employee

{

public int Id { get; set; }

public string Name { get; set; } = null!;

public string Email { get; set; } = null!;

public int DeptId { get; set; }

public DateTime DateOfBirth { get; set; }

public int Salary { get; set; }

public virtual Dept Dept { get; set; } = null!;

}

**Note:** The Dept ↔ Employee relationship is correctly represented using navigation properties.

**Step 5: Configure DbContext**

appdbContext.cs should include:

public partial class appdbContext : DbContext

{

public appdbContext() { }

public appdbContext(DbContextOptions<appdbContext> options) : base(options) { }

public virtual DbSet<Dept> Depts { get; set; } = null!;

public virtual DbSet<Employee> Employees { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer("User ID=sa;password=examlyMssql@123;Server=localhost;Database=appdb;Trusted\_Connection=False;Persist Security Info=False;Encrypt=False");

}

}

}

**Step 6: Create Controller to Fetch Data**

Controllers/EmployeeController.cs:

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using EmployeeDeptApp.Models;

namespace EmployeeDeptApp.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class EmployeeController : ControllerBase

{

private readonly appdbContext \_context;

public EmployeeController(appdbContext context)

{

\_context = context;

}

// GET: api/Employee

[HttpGet]

public async Task<ActionResult<IEnumerable<Employee>>> GetEmployees()

{

return await \_context.Employees.Include(e => e.Dept).ToListAsync();

}

// GET: api/Employee/1

[HttpGet("{id}")]

public async Task<ActionResult<Employee>> GetEmployee(int id)

{

var emp = await \_context.Employees.Include(e => e.Dept).FirstOrDefaultAsync(e => e.Id == id);

if (emp == null) return NotFound();

return emp;

}

}

}

**Step 7: Run and Test**

1. Restore packages:

dotnet restore

1. Build project:

dotnet build

1. Run project:

dotnet run

* Use Postman or Swagger (https://localhost:5001/swagger) to test endpoints.
* GET /api/Employee will display all employees along with their department data.