1. Getting employee details

private static void GetAllEmployeeDetails\_Input()

{

var ctx = new EFCoreTrainingContext();

try

{

var emps = ctx.Set<Emp>().FromSqlRaw("GetEmployeeDetailsByDeptNo @p0", 30).ToList();

foreach (var e in emps)

{

Console.WriteLine($"{e.Ename} {e.Sal}");

}

}

catch (Exception ex)

{

Console.WriteLine(ex.InnerException.ToString());

}

}

1. Add and Delete employee in Disconnected architecture.

private static void DisconnectedAchitecture()

{

var ctx = new AugETG\_2021Context();

try

{

Emp employee = new Emp();

ctx.Dispose();

AddNewEmployee(employee);

//--------------------------------------------------------------------------------------

var emp = ctx.Emps.Where(x => x.Empno == 1234).SingleOrDefault();

ctx.Dispose();

DeleteEmployeeByEmpNo(emp);

}

catch(Exception ex)

{

Console.WriteLine(ex.Message);

}

}

private static void DeleteEmployeeByEmpNo(Emp emp)

{

var ctx = new AugETG\_2021Context();

Console.WriteLine(ctx.Entry(emp).State.ToString());

ctx.Attach(emp).State = EntityState.Deleted;

ctx.SaveChanges();

Console.WriteLine("Employee deleted");

}

private static void AddNewEmployee(Emp employee)

{

employee.Empno = 1234;

employee.Deptno = 20;

var ctx = new AugETG\_2021Context();

Console.WriteLine(ctx.Entry(employee).State.ToString());

ctx.Attach(employee).State = EntityState.Added;

ctx.SaveChanges();

Console.WriteLine("New employee added");

}

1. Add new customer with multiple orders.

private static void AddnewcustomerAndMultipleOrder()

{

var ctx = new Context();

Customer newcust = new Customer();

newcust.ID = 3;

newcust.Name = "Abc";

Order ord = new Order();

ord.Order\_ID = 16;

ord.Amount = 1000;

ord.OrderDate = DateTime.Now;

Order ord2 = new Order();

ord2.Order\_ID = 17;

ord2.Amount = 2000;

ord2.OrderDate = DateTime.Now;

List<Order> myorders = new List<Order>();

myorders.Add(ord);

myorders.Add(ord2);

newcust.Orders = myorders;

try

{

ctx.Customers.Add(newcust);

ctx.SaveChanges();

Console.WriteLine("Customer and multiple order is created");

}

catch (Exception ex)

{

Console.WriteLine(ex.InnerException.ToString());

}

}

1. Adding the orders for existing customer.

private static void AddOrder()

{

var ctx = new Context();

var custID=ctx.Customers.Where(c=>c.ID==2).SingleOrDefault();

Order ord = new Order();

ord.Order\_ID = 2;

ord.Amount = 1000;

ord.OrderDate = DateTime.Now;

ord.cust = custID;

try

{

ctx.Orders.Add(ord);

ctx.SaveChanges();

Console.WriteLine("order is created");

}

catch (Exception ex)

{

Console.WriteLine(ex.InnerException.ToString());

}

}

1. Deleting order with order ID

private static void DelOrder()

{

var ctx = new Context();

var removeRow = ctx.Orders.SingleOrDefault(o => o.Order\_ID == 1);

try

{

ctx.Orders.Remove(removeRow);

ctx.SaveChanges();

Console.WriteLine("Order deleted successfully");

}

catch (Exception ex)

{

Console.WriteLine(ex.InnerException.ToString());

}

}

1. Deleting all the orders linked to customer ID.

private static void DelOrderByCustID()

{

var ctx = new Context();

var customerID = ctx.Customers.SingleOrDefault(c => c.ID == 3);

var orders = ctx.Orders.Where(o => o.cust == customerID);

try

{

foreach (var order in orders)

{

ctx.Orders.Remove(order);

}

ctx.SaveChanges();

Console.WriteLine("Deleted the orders for given customer ID");

}

catch(Exception ex)

{

Console.WriteLine(ex.InnerException.ToString());

}

}

1. Get the count of orders placed by each customer

private static void CountByCustID()

{

var ctx = new Context();

try

{

var ctx = new Context();

var numorders = ctx.Orders.GroupBy(o => o.cust.ID).Select(n => new

{

CustomerName = n.Key,

Count = n.Count()}).ToList();

}).ToList();

/\*var numorders = from o in ctx.Orders

group by o.\*/

foreach (var item in numorders)

{

Console.WriteLine(item.CustomerName + " " + item.Count);

}

}

catch (Exception ex)

{

Console.WriteLine(ex.InnerException.ToString());

}

}

1. School and Student classes

public class School

{

[Key, DatabaseGenerated(DatabaseGeneratedOption.None)]

public int SchoolID { get; set; }

public string SchoolName { get; set; }

public string Address { get; set; }

public virtual ICollection<Student> Students { get; set; } //navigation prop

}

public class Student

{

[Key, DatabaseGenerated(DatabaseGeneratedOption.None)]

public int Student\_ID { get; set; }

public string Student\_Name { get; set; }

public int Age { get; set; }

public int Grade { get; set; }

public School school { get; set; } //navigation prop

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

//Property Configurations

//Fluent API

modelBuilder.Entity<School>()

.Property(s => s.SchoolName)

.HasMaxLength(30)

.IsRequired();

modelBuilder.Entity<School>()

.Property(s => s.Address)

.IsRequired()

.HasMaxLength(100);

}