في موضوع entity framework
لازم نعرف انه هناك نوعين عند الأتصال بقاعدة البيانات وهما 1:مبدا الداتا بيس اولا,2:مبدا الكود اولا

مبدأ الكود أولا هو أننا نقوم بكتابة الكلاسات الخاصه بالجدوال التي ستكون في قواعد البيانات لاحقا نكتبها بلغة السي شارب مثلا وبعدين نقوم بتحويلها الى جداول في قواعد البيانات بعد التأكد منها مع أمكانية تعديلها
لفعل هذا الشي نقوم بتنزيل مكتبتين مهمتين وهما microsoft.entityframeworkcore.sqlserver:1
2:microsoft.entityframeworkcore.tools.

طيب لنفترض اننا قمنا بعمل كلاس خاص بالطلاب واردنا ان نقوم بتحويله الى جدول في الداتا بيس اول خطوة نقوم بإنشاء نص الاتصال بقاعدة البيانات وافضل طريقة لعملة هو اننا نقوم بعمل ستاتيك كلاس ويكون خاص
بنصوص الاتصال بقواعد البيانات مثل هذا

```
public static  class Connections
  {
    public const string sqlConStr = "server=.;" +
      "Database=dbEfTest;User Id =sa; password=*******;TrustServerCertificate=true;";

  }
```

في مكان الداتابيس هنا عادي لو كانت قاعدة البيانات موجودة فعلا او غير موجوده لانه كانت غير موجودة هو سينشئ قاعدة بيانات جديدة بنفس هذا الاسم اما باقي البيانات من اسم السيرفر واليوسر والباسوورد لازم يطابق
بيانات جهازك

الان نسوي الكلاس المسؤول عن ادراة الشغل بينك وبين قواعد البيانات وبيكون كالتالي

```
public class AppDbContext : DbContext
  {
    protected override void OnConfiguring(DbContextOptionsBuilder option)
    {
      option.UseSqlServer(Connections.sqlConStr);
    }

    public DbSet<Student> Students { get; set; }

  }
```

نلاحظ انه الكلاس يورث من كلاس اسمه دي بي كونتيكس وهذا الكلاس موجود في مكتبة انتيتي فريموورك كور والشي الثاني قمنا

بعمل اوفر رايد للفنكشن اون كونفيجيورينج نمرر لها نفس البارامیتر الموجود علاه
وبداخلها نكتب كود الاتصال مع قواعد البيانات وبهذه الطريقة بيكون تعاملنا مع قاعدة البيانات حسب النص الي مررناه داخل فنكشن يوز اس كيو ال سيرفير
وبعد هذه الفنكشن نسوي دي بي سيت للكلاس الي نبغا تحويلة الى قاعدة البيانات حسب الكود اعلاه طيب الان ضبطنا كلاس الدي بي كونتيست الخطوة الثانيه نعمل مايجراشن زي هذا الامر

add-migration تكتب اسم من عندك للمايجراشن

لما نسوي هذا الامر بينشئ عندنا كلاس يحتوي على هذه الاشياء

```
public partial class studentTable : Migration
  {
    protected override void Up(MigrationBuilder migrationBuilder)
    {
      migrationBuilder.CreateTable(
        name: "Students",
        columns: table => new
        {
          Id = table.Column<int>(type: "int", nullable: false)
            .Annotation("SqlServer:Identity", "1, 1"),
          Name = table.Column<string>(type: "nvarchar(max)", nullable: false),
          Email = table.Column<string>(type: "nvarchar(max)", nullable: false),
          Age = table.Column<int>(type: "int", nullable: false),
          Grade = table.Column<int>(type: "int", nullable: false),
          Birthdate = table.Column<DateTime>(type: "datetime2", nullable: false)
        },
        constraints: table =>
        {
          table.PrimaryKey("PK_Students", x => x.Id);
        });
    }

    protected override void Down(MigrationBuilder migrationBuilder)
    {
      migrationBuilder.DropTable(
        name: "Students");
    }
  }
```

studentTable هو اسم المايجراشن الي قمنا بإنشاءة

لو اردنا التراجع عن هذا الامر بدون عمل تحديث لقاعدة البيانات نعمل هذا الامر

remove-migration

طبعا تنفيذ المايجراشن يكون بالتسلسل فلو عندي اكثر من مايجراشن وعملت الامر السابق بيحذف اخر مايجراشن تم عمله
طيب الان نريد يتم انشاء الجدول فعليا في الداتا بيس نعمل هذا الامر

update-database

وزي ما قلنا الامر متسلسل لو عملت اكثر من مايجراشن بدون تحديث للداتا بيس وجيت سويت تحديث رح يسوي الفنكش الي اسمها أب في جميع المايجراشنات الي سويتها وهي هذه الفنكشن

```
protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "Students",
            columns: table => new
            {
                Id = table.Column<int>(type: "int", nullable: false)
                    .Annotation("SqlServer:Identity", "1, 1"),
                Name = table.Column<string>(type: "nvarchar(max)", nullable: false),
                Email = table.Column<string>(type: "nvarchar(max)", nullable: false),
                Age = table.Column<int>(type: "int", nullable: false),
                Grade = table.Column<int>(type: "int", nullable: false),
                Birthdate = table.Column<DateTime>(type: "datetime2", nullable: false)
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_Students", x => x.Id);
            });
    }
```

طيب الان سويت الجدوال داخل قاعدة البيانات واردت اتراجع عن احد الجدوال او اي تعديل خاص باحد الجداول ايش الحل نسوي هذا الامر

update-databse -migration:هنا نكتب اسم المايجراشن المراد التراجع عنه

الان هذا الامر لو سويناه رح يروح لكلاس المايجراشن الي شرحناه سابقا ورح ينفذ الفنكشن الي اسمها داون وهي هذه الفنكشن

```
 protected override void Down(MigrationBuilder migrationBuilder)
    {
```

```
        migrationBuilder.DropTable(
            name: "Students");
    }
```

طيب عندي سؤال مهم لو انا قلتلك لدي اربعه كلاسات مايجراشن وهي خاصه بانشاء اربعه جداول في قاعدة البيانات مرتبة حسب الاتي في ملف المايجراشن

193712792StudentTable.cs
1326169612Department.cs
129636636Course.cs
971397629Grades.cs

وانا قلت لك اريدك تتراجع عن انشاء جدول الاقسام الي هو ترتيب المايجراشن حقه هو الثاني
شي طبيعي انك انت بتقولي سهله بس نسوي هذا الامر

```
update-database -migration:1326169612Department
```

لاكن المفاجاة انه هذا الامر تعرف ايش يسوي؟
يقوم بالتراجع عن جميع تعديلات قاعدة البيانات التي حدثت من بعد المايجراشن الي انت حددته بمعنى اخر انه يحذف جدول الدرجات وجدول الكورسات
لانه زي ماقلنا سابقا انه التنفيذ يكون متسلسل طيب والحل ايش نسوي ؟
الحل عندي يقولك فيه طريقة انك تعمل مايجراشن عكسي اكيد مافهمت خلني اشرح لك خطوة بخطوة انت ايش هدفك
هدفك هو انك تتراجع عن انشاء جدول الاقسام بدون م تتأثر جداول الكورسات والدرجات
يقولك تروح تعمل مايجراشن جديد وتسميه مثلا حذف جدول الاقسام لاكن بشرط انه كلاس الدي بي كونتيكيس الكلاسات الي عملتلها دي بي سيت قد سويت لها مايجراشن يعني الكلاس في حالة استقرار وكذلك قاعدة البيانات
فلما تعمل المايجراشن الجديد رح ينشئ كلاس زي كلاس المايجراشن الي شرحناه سابقا ولاكن يختلف عنه انه يسوي لك فنكشن الأب وفنكشن الداون فاضيات مافيهن كود وانت حط الكود الي تبغاه زي هكذ

```
protected override void Up(MigrationBuilder migrationBuilder)
    {

    }

  protected override void Down(MigrationBuilder migrationBuilder)
    {

    }
```

صح احنا قلنا سابقا انه لما نعمل امر التحديث يروح ينفذ فنكشن الأب صح
خلاص انحلت ايش تسوي تروح لفنكشن الأب في ملف المايجراشن الجديد ونكتب هكذا الكود زي ماهو موضح ادناه

```
protected override void UP(MigrationBuilder migrationBuilder)
    {
       migrationBuilder.DropTable(
          name: "Departments");
    }
```

طبعا احنا سوينا هذا الكود بالذات لانه هدفنا من هذا المايجراشن الجديد هو حذف ملف الاقسام انت لو كان تبغى تعدل في جدول او اي تعديل في قاعدة البيانات تكتب الكود الي انت تبغا الهدف منه داخل فنكشن الأب

طيب وبعدين نسوي هذا الامر عشان تتنفذ كل هذه الامور

```
update-database
```

من خلال الامر هو تلقائيا بيروح لاخر مايجراشن وبينفذ فنكشن الأب الخاصه به الي كتبناها سابقا وهذا هو الحل ي بشمبرمج

بعد م خلصنا موضوع المايجراشن ندخل موضوع جديد وهو
وهو لو بغيت بسوي علاقة ون تو ون بين كلاسين ايش نسوي لازم نحدد عمود البرايماري كي في الجدول الأب الي هو في مثالنا هذا جدول الطلاب وثاني شي نعمل نافيجاشن بروبيرتي اتريبيوت وهو انه نسوي
اتريبيوت جديد في الكلاس نوعه من نوع الكلاس الابن والي هو في مثالنا هذا هو كلاس الدرجات
لاحظ انه اخر اتريبيوت في كلاس الطلاب

```
public class Student
  {
     [Key]
     public int Id { get; set; }
     public string Name { get; set; }
     public string Email { get; set; }
     public int Age { get; set; }
     public int Grade { get; set; }
     public DateTime Birthdate { get; set; }
     public Grade grade { get; set; }
  }
```

طيب نجي لكلاس الابن الي هو الدرجات ونعمل نفس السابق بس نضيف نحدد عمود الفوريجن كيي زي المثال ادناه

```
public class Grade
  {
     [Key]
     public int Id { get; set; }
     public decimal physics { get; set; }
```

```
    public decimal chemistry { get; set; }
    public decimal programming { get; set; }

    [ForeignKey("student")]
    public int studentId { get; set; }
    public Student student { get; set; }
  }
```

وبعدين نعمل مايجراشن زي كل مره وراح تنعمل العلاقة في قاعدة البيانات
لاحظ انه هذه الاشياء الي استخدمناه فوق عمود الاي دي وعمود الفوريجن كيي وهما هذولا
[Key],[ForeignKey("student")]
هذولا اسمهم داتا انوتاشن ونستخدمهم عن طريق هذه المكتبة ادناه

```
using System.ComponentModel.DataAnnotations;
```

بإمكانك تبحث عنهم أكثر

طيب نجي الان لعلاقة متعدد الى واحد نعمل نفس ماعملنا سابقا لاكن نضيف انه في الجدول الذي يحتوي على علاقة الكثير نعمل فيه اتريبيوت من نوع اي كوليكشن وياخذ الداتا تايب للكلاس الذي حتوي على علاقة الواحد
مثلا عندنا علاقة واحد الى متعدد بين كلاس الاقسام والطلاب الجدول الى يحتوي على علاقة الكثير هو كلاس الاقسام نعمل فيه هكذا

```
 public class Department
  {
    public int Id { get; set; }
    public string Name { get; set; }
    public ICollection<Student> Students { get; set; }

  }
```

والكلاس الي يحتوي على علاقة الواحد هو كلاس الطلاب ونعمل فيه هكذا

```
public class Student
  {
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public int Age { get; set; }
    public int Grade { get; set; }
    public DateTime Birthdate { get; set; }
    public Grade grade { get; set; }
```

```
    [ForeignKey("department")]
    public int departmentId { get; set; }
    public Department department { get; set; }

}
```

اما بالنسبة لعلاقة المتعدد الى متعدد معروفه انها تحتاج جدول وسيط فلو عندنا علاقة متعدد الى متعدد بين الطلاب والكتب بيؤدي اننا ننشئ كلاس جديد اسمه كتب الطلاب وهو الوسيط بينهم فبيكون شكل كلاس الطلاب هكذا

```
public class Student
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public int Age { get; set; }
    public int Grade { get; set; }
    public DateTime Birthdate { get; set; }
    public Grade grade { get; set; }

    [ForeignKey("department")]
    public int departmentId { get; set; }
    public Department department { get; set; }

    public ICollection<StudentBook> books { get; set; }
}
```

وبيكون شكل كلاس الكتب هكذا

```
public class Book
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public string Author { get; set; }
    public DateTime Created { get; set; }

    public ICollection<StudentBook> Students { get; set; }
}
```

اما بالنسبة لشكل الكلاس الوسيط فبيكون هكذا

```
public class StudentBook
  {
     public int Id { get; set; }

     [ForeignKey("student")]
     public int studentId { get; set; }
     public Student student { get; set; }

     [ForeignKey("book")]
     public int bookId { get; set; }
     public Book book { get; set; }

     public DateTime getDate { get; set; }
  }
```

الان بالنسبة لعمليات الاضافة والتعديل والحذف عند اضافة ريكورد او سطر جديد في الداتا بيس في احد الجداول لازم اول حاجه نعمل اوبجيكت من كلاس الدي بي كونتيكتس
وبعدين نسوي اوبجيكت من نوع الكلاس الي احنا نضيف فيه السطر احنا في مثالنا هذا رح نضيف سطر في جدول الاقسام زي الاتي

```
using var db = new AppDbContext();

var department = new Department()
{
   Name = "Ahmed"
};

db.Departments.Add(department);
db.SaveChanges();
```

اما بالنسبة لتعديل صف معين في الداتا بيس اول شي لازم نبحث على الصف المراد تعديله في الجدول عن طريق احد فنكشن اللينكيو وفي مثالنا هذا استخدمنا دالة الفايند

```
using var db = new AppDbContext();

var department = db.Departments.Find(3);

if(department != null)
{
   department.Name = "Ali";
```

```
    db.SaveChanges();
}
```

اما بالنسبة للحذف لا يختلف كثيرا عن التعديل

```
using var db = new AppDbContext();

var department = db.Departments.Find(3);


if(department != null)
{
    db.Remove(department);
    db.SaveChanges();
}
```

الان لدينا بعض استخدامات الداتا نوتاشن في توجيه البروبيرتيز في الكلاس والتحكم في الاخطاء مثلا زي نوتاشن الريكوايريد معناه انه هذا العمود مطلوب
وكذلك نوتاشن الايرور يعطي المستخدم نظرة عن الخطأ الواقع فيه وكذلك نوتاشن لاعطاء النص حد معين زي المثال ادناه

```
 public class Department
    {
        [Key]
        public int Id { get; set; }

        [Required(ErrorMessage ="Plese Enter the Name")]
        public string Name { get; set; }

        [MaxLength(5,ErrorMessage ="Max len can't be > 5 chrs")]
        public string? des {  get; set; }

        public ICollection<Student> students { get; set; }
    }
```

طبعا هذي الامور لو تبغاها تتنفذ في الداتا بيس لازم تعمل مايجراشن
وكذلك حتى تقدر تستخدم هذي الامور وتستفيد منها في البرنامج لازم في كلاس البروجرام تعمل هكذا

```
using var db = new AppDbContext();

// Insert
var department = new Department()
```

```
{
   Name = "Ahmed 02",
   des="1234"
};

var context = new ValidationContext(department);
var errors = new List<ValidationResult>();

if (!Validator.TryValidateObject(department, context, errors, true))
{
   foreach (var validationResult in errors)
   {
      Console.WriteLine(validationResult);
   }
}

else
{
   db.Departments.Add(department);
   db.SaveChanges();
}
```

طيب الان يقولك فيه حاجه خاصه بقاعدة البيانات وهو افترض لو عندك جدول الاقسام وفيه بيانات وعندك جدول الطلاب فيه بيانات ومربوط بجدول الاقسام عن طريق فورين كيي وجيت تبغى تحذف جدول الاقسام
هنا يقولك قواعد البيانات اعطتك اربعه انواع
اول نوع يقولك لما تحذف جدول الاقسام هو تلقائيا سيحذف البيانات المتعلقة به في جدول الطلاب وهذه الطريقة تسمى الكاسكاد
النوع الثاني انه لا يتم حذف جدول الاقسام الا لما تروح انت بنفسك وتحذف جميع بيانات الاقسام في الطلاب بعدين هو يحذف جدول الاقسام وهذه الطريق تسمى ريستريكت
النوع الثالث هو يخليك براحتك وهو انك تحذف جدول الاقسام براحتك ولاكن بياناته تضل في جدول الطلاب والحياه حلوه مع انه غير صحيح ولاكنه موجود وتسمى هذه الطريقه نوو اكشن
الطريقة الرابعه انها تروح للفورين كيي الموجود في جدول الطلاب الخاص بالاقسام يعطيه قيمة النل وهذه الطريقة تسمى سيت نل

طيب ايقولك انه الديفولت الي يتم انشاءة تلقائيا في المايجراشن هو نوع الكاسكاد وهذا خطير لانك لو حذفت الفورين كيي بينحذف كل الصفوف الاساسية في جدول البرايماري كي
طيب احنا من وين نغيره يقولك انت لما تعمل مايجراشن قبل م تعمل تحديث تروح لملف المايجراشن وتروح للحاجه الي مكتوب عليها كونستراينت او اد فوريين كيي زي كذا

```
constraints: table =>
            {
               table.PrimaryKey("PK_StudentBooks", x => x.Id);
               table.ForeignKey(
                  name: "FK_StudentBooks_Books_bookId",
                  column: x => x.bookId,
```

```
            principalTable: "Books",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
        table.ForeignKey(
            name: "FK_StudentBooks_Students_studentId",
            column: x => x.studentId,
            principalTable: "Students",
            principalColumn: "Id",
            onDelete: ReferentialAction.Cascade);
    });
```

او

```
migrationBuilder.AddForeignKey(
        name: "FK_Students_Departments_departmentId",
        table: "Students",
        column: "departmentId",
        principalTable: "Departments",
        principalColumn: "Id",
        onDelete: ReferentialAction.Cascade);
```

لاحظ اخر حاجه مكتوب اون ديليت تمسح الكاسكاد وتكتب النقطة وبتطلع لك جميع الانواع الي شرحناها سابقا واختار الي بغيته منها

هنا فيه حاجه لازم تعرفها وهي انه يقولك لما تعمل هذي التعديلات في ملف المايجراشن بمجرد مسح المايجراشن تنمحي التعديلات وترجع تكتبها من جديد ف يقولك الحل الافضل هو انك تعمل
الحاجات الي تشوف انها عامه بدل م تسويها في كل مايجراشن تروح تسويها مره وحده في ملف الدي بي كونتيكتس الي هو له مسمى ثاني وهو الفلونت اي بي ايي بمجرد كتابته هنا
رح يتم عمله في كل مايجراشن تلقائيا فنروح للدي بي كونتيكتس ونسوي هذا الكود

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Department>()
            .HasMany(p => p.Students)
            .WithOne(d => d.department)
            .OnDelete(DeleteBehavior.Restrict);
    }
```

طبعا هذي الدالة من اسمها وهي تعمل عند انشاء الموديلات في المايجراشن والكود الي فيها كانها تقولها جدول الاقسام يحتوي على الكثير من جدول الطلاب
وجدول الطلاب يحتوي على واحد من الاقسام واخر سطر معروف شرحه

لاحظ انه الكود السابق خاص بعلاقة واحده وهي الطلاب والاقسام اذا تبغى العلاقات الباقية تكتبها نفس الطريقة لاكن هذا مش طريقة فعاله فيقولك فيه كود تكتبه هو بنفسه يروح على كل العلاقات كلها ويخليها من نوع ريستريكت

زي هكذا

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
       foreach(var relationShip in modelBuilder.Model.GetEntityTypes()
          .SelectMany(e => e.GetForeignKeys()))
       {
          relationShip.DeleteBehavior = DeleteBehavior.Restrict;
       }
    }
```

فيه ملاحظه جديدة يقولك لو سويت جدول جديد ومعه علاقة مع جدول سابق والجدول السابق مكتوب في كلاس الدي بي كونتيكست عن طريق الدي بي سيت
لاكن الجديد مش مكتوب داخل الدي بي كونتيكست ولما تعمل مايجراشن يروح يضيف الجدول الجديد الى الداتا بيس مع انه ماشي له دي بي سيت بسبب انه مع علاقة مع جدول سابق موجود في الدي بي كونتيكست
طيب اعتبر معنا حالة نفس الماضية لاكن بغيت لما اعمل مايجراشن الجدول الجديد ما يتم عمله في الداتا بيس ايش نسوي نروح للجدول السابق وفوق الاتريبيوت نافيجاشن تبع الجدول
الجديد نكتب داتا انوتاشن اسمها نوت ماب زي كذا

```
[NotMapped]
public ICollection<StudentBook> books { get; set; }
```

وهذي الطريقة تستخدم حتى مع الاعمده العادية الي م نبغاها تضهر في الداتا بيس ولاكنها تظهر على مستوي الانتيتي فريمووورك عن نوتاشن النوت مابيد

هذه طريقة وفيه طريقة ثانية عن طريق الفلونت اي بي ايي

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
       modelBuilder.Ignore<Attendance>();
    }
```

طيب افترض معنا جدول جديد ليس لديه علاقة مع اي جدول وبنفس الوقت انا ما ابغى يكون له دي بي سييت في كلاس الدي بي كونتيكست ولاكن ابغاه يروح ع الداتا بيس لما اسوي مجراشن ايش تسوي
يقولك بس تروح تكتب هذا الكود في كلاس الدي بي كونتيكست

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
```

```
        modelBuilder.Entity<اسم الجدول>();
    }
```

طيب افترض انه معك جدول انت سويت عليه تعديلات ومايجراشن والى اخره لاكن بعد فتره حبيت توقف التعديلات على هذا الجدول
على مستوى الداتا بيس وليس على مستوى الاينتيتي فريموورك
يعني تبغى تعدل على الجدول في الاينتيتي فريموورك براحتك لاكن التعديل م تبغاه يتحول على الداتا بيس لما تسوي مايجراشن
هذي حلها سهل بس تروح على الدي بي كونتيكتس وتسوي هذا الكود

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<اسم الجدول>().ToTable("هنا تكتب اسم الجدول في قاعدة البيانات", a =>
a.ExcludeFromMigrations());
    }
```

طيب يقولك لو بغيت اسم الجدول يتغير في الداتا بيس يعني مش نفس اسم المودل او الكلاس البرنامج يقولك بس تروح تعمل تعمل داتا
انوتاشن فوق اسم الجدول بهذا الشكل
طبعا حتى العمود ينطبق عليه نفس الشي

```
[Table("std",Schema ="sss")]
  public class Student
  {
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    [Column("the age",TypeName ="varchar(20)")]
    public int Age { get; set; }
    public int Grade { get; set; }
    public DateTime Birthdate { get; set; }
    public Grade grade { get; set; }

    [ForeignKey("department")]
    public int departmentId { get; set; }
    public Department department { get; set; }

    public ICollection<StudentBook> books { get; set; }
  }
```

لاحظ انه كلمة السكيما في نوتشن التابل وكلمة التايب نام في نوتاشن العمود هي اختيارية مش بالضرورة تكتبها
طيب هنا عملناهم بطريقة الداتا نوتاشن خلونا نعملهم بطريقة الفلونت اي بي اي زي كذا

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Attendance>().ToTable("myAtt", "mysc");

        modelBuilder.Entity<Attendance>().Property(x => x.DayName)
            .HasMaxLength(14);

        modelBuilder.Entity<Attendance>().Property(x => x.name)
            .HasColumnName("theName")
            .HasColumnType("varchar(50)");

        modelBuilder.Entity<Attendance>().Ignore(x => x.theData);
    }
```

طيب الان معنا موضوع انه لو عندي اعمده قيمتها مستندة من اعمدة اخرى زي هذا المثال

```
public class Invoice
 {
    [Key]
    public int Id { get; set; }

    public string customerTitle { get; set; }

    public string customerName { get; set; }

    public string fullName { get; set; }

    public decimal price { get; set; }
    public decimal qty { get; set; }

    public decimal total { get; set; }

    public DateTime createdDate { get; set; }
 }
```

لاحظ عندك عمود الفل نام وعمود التوتال قيمة الفل نام عبارة عن عمود الكستمر تايتل بالاضافة الى قيمة عمود الكستمر نام
وقيمة عمود التوتال عبارة عن عمود البرايس مضروبا في عمود الكمية
هنا يقولك عشان يعرف الانتيتي فريموورك هدفك هذا لازم تقوم بكتابته في الفلونت اي بي اي داخل الدي بي كونتيكتس زي هكذا

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
```

```
        modelBuilder.Entity<Invoice>().Property(x => x.qty)
                .HasDefaultValue(1);

        modelBuilder.Entity<Invoice>().Property(x => x.createdDate)
                .HasDefaultValueSql("GETDATE()");

        modelBuilder.Entity<Invoice>().Property(x => x.fullName)
                .HasComputedColumnSql("[customerTitle] + ' ' + [customerName]");
        modelBuilder.Entity<Invoice>().Property(x => x.total)
               .HasComputedColumnSql("[price] * [qty]");
    }
```

طيب الحين معنا موضوع كيف نسوي انديكس في الجدول سواء كان على عمود واحد او اكثر من عمود عندك المثال ادناه

```
[Index("اسم العمود الاول","...")]
public class Student
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public int Age { get; set; }
    public int Grade { get; set; }
    public DateTime Birthdate { get; set; }
    public Grade grade { get; set; }

    [ForeignKey("department")]
    public int departmentId { get; set; }
    public Department department { get; set; }

    public ICollection<StudentBook> books { get; set; }

    //[NotMapped]
    //public ICollection<Attendance> attendances { get; set; }
}
```

طبعا هنا سويناه بطريقة الداتا نوتاشن
وهذا المثال ادناه بطريقة الفلونت اي بي اي

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>().HasIndex(x => x.Name);
```

```
    }
```

طيب هنا يقولك فيما يتعلق بالانديكس يفضل انك تخليه يونيك اي فريد وكذلك يفضل او براحتك انك تضيف له فيلتر شوف المثال ادناه

```
[Index("اسم العمود",IsUnique = true , Name="الاسم الي انت تبغاه")]
public class Student
{
    [Key]
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public int Age { get; set; }
    public int Grade { get; set; }
    public DateTime Birthdate { get; set; }
    public Grade grade { get; set; }

    [ForeignKey("department")]
    public int departmentId { get; set; }
    public Department department { get; set; }

    public ICollection<StudentBook> books { get; set; }

    //[NotMapped]
    //public ICollection<Attendance> attendances { get; set; }
}
```

هنا نقدر نعدل حتى على اسم الانديكس الي نبغاه يكون في قاعدة البيانات

طيب بطريقة الفلونت اي بي اي نضيف فيلتر ونخليه يونيك

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.Entity<Student>().HasIndex(x =>
x.Name).IsUnique().HasDatabaseName("الاسم الي انت تبغاه");
    }
```

الان معنا موضوع الاعمدة التسلسلية وهي عبارة عن عدة اعمدة في جداول مختلفة الزيادة فيها تكون تسلسلية مثلا لو في الجدول الاول اضاف عمود يستخدم التسلسل
وضاف فيه بيانات ثم راح للجدول الثاني وضاف بيانات العمود الذي يحتوي على التسلسل البيانات الي بتكون فيه بتبدأ من اخر قيمة الي اخذها الجدول الاول
هذا نادرا يستخدم الا في الانظمة المصرفية واذا م فهمت روح ابحث اكثر
طيب كيف نسويه في الاينتيتي فريموورك نروح للفلونت اي بي اي ونكتب هذا الكود

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.HasSequence<int>("DeliveryOrder")
            .StartsAt(101)
            .IncrementsBy(1);

        modelBuilder.Entity<Book>().Property(p => p.DeliveryOrder)
            .HasDefaultValueSql("Next Value For DeliveryOrder");

        modelBuilder.Entity<Uniform>().Property(p => p.DeliveryOrder)
            .HasDefaultValueSql("Next Value For DeliveryOrder");
    }
```

نلاحظ اننا في البداية قمنا بتعريف التسلسل الي بيتم مشاركته مع الاعمدة
وثاني خطوة حددنا الاعمدة الي بتؤخذ قيمها من هذا التسلسل

طيب الان معنا موضوع انك كيف تأخذ سكريبت او ملف نصي لجميع المايجراشن الي سويتهم وتلصقهم في الداتا بيس باعتبارك سويته بلغة اس كيو ال سيرفر والطريقة هيا انك تروح
على الكونسول وتكتب هذا الامر

```
script-migration
```

اما اذا تبغى سكربت خاص ب مايجراشن لحالها لازم تكتب كذا

```
script-migration قبل الاسم
```

حيث كلمة قبل تمثل اسم المايجراشن الي قبل المايجراشن المطلوب وكلمة اسم تمثل اسم المايجراشن المطلوب

اليوم معنا موضوع مهم وهو مبدأ قاعدة البيانات اولا
طيب كيف نعمل ساكفولد لقاعدة البيانات على البرنامج تبعنا ترمح على الكونسول وتكتب هذا الامر

```
scaffold-dbcontext 'هنا نكتب نص الاتصال بقاعدة البيانات' Microsoft.EntityFrameworkcore.SqlServer
-OutputDir Models -contextDir Data
-DataAnnotations -table
```

طبعا احنا سوينا هذا النص ادناه عشان نحدد البروفايدر

```
Microsoft.EntityFrameworkcore.SqlServer
```

وسوينا هذا النص ادناه لانه تلقائيا ينشأ لك الكلاسات في البرنامج بدون ترتيب او تنظيم فاحنا قلنا له انه كلاسات الموديلس حطها في مجلد اسمه موديلس

```
-OutputDir Models
```

وسوينا هذا النص عشان نقولة انه كلاس الدي بي كونتيكتس حطه في مجلد اسمه داتا

-contextDir Data

وسوينا هذا النص ادناه عشان الانتيتي فريموورك تسويلك اغلب خصائص الاعمدة في الجداول او الموديلات بطريقة الفلونت اي بي اي وتمليلك كلاس الدي بي كونتيكس بحاجات كثير ممكن اننا نعملها بطريقة
الانوتاشن فوق اسامي الاعمدة عشان الترتيب فنكتب له هذا النص ادناه

-DataAnnotations

وسوينا هذه النص ادناه عشان نحدد الجداول الي نبغا نتعامل معها ومانبغى لها مودل طبعا اذا كنت تبغى جميع الجداول ماشي داعي تكتب هذا النص

-table هنا تكتب اسامي الجداول مع فاصلة بينهم

هذا الامر لو كتبناه رح يجيب كل المايجراشن الي سويناهم بدون م نروح لقاعدة البيانات وهذا الامر هو

get-migration

طبعا الان موضوعنا على انك كيف بداية م تنشأ الجدوال او تعمل لها مايجراشن تحط فيها قيم مبدئية بمعنى اخر صفوف سهله نروح على الفلونت اي بي اي ونكتب هذا الكود

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
    {

        modelBuilder.Entity<اسم الجدول>().HasData(
            new Gender() { Id = 1, genderName = "Male" },
            new Gender() { Id = 2, genderName = "Female"}
            );

    }
```

ملاحظه هنا حتو لو كان الاي دي انت مسويه اوتو انكريمينت الزيادة فيه تكون تلقائية لازم تكتبه هنا في البيانات

الان معنا حاجه حلووه على كيف كيفك افترض انه معك جدولين معهم علاقة وهم مثلا الطلاب والاقسام وانت تبغى تضيف لقاعدة البيانات الطلاب والاقسام بنفس الوقت شوف المثال ادناه

```
var db = new AppDbContext();
```

```
var departments = new List<Department>() {
   new Department()
{
   Name = "list 1", des = "xyz" ,
   students = new List<Student>() {
     new Student() { Name = "S1.1", Email = "S@s.com", Age = 10 , Grade = 10, Birthdate =
DateTime.Now },
     new Student() { Name = "S2.1", Email = "S2@s.com", Age = 11 , Grade = 11, Birthdate =
DateTime.Now },
   }
},
      new Department()
{
   Name = "list 2", des = "qaz" ,
   students = new List<Student>() {
     new Student() { Name = "S1.2", Email = "S@s.com", Age = 10 , Grade = 10, Birthdate =
DateTime.Now },
     new Student() { Name = "S2.2", Email = "S2@s.com", Age = 11 , Grade = 11, Birthdate =
DateTime.Now },
   }
},

   };


db.Departments.AddRange(departments);
db.SaveChanges();
```

لاحظ انه كل طالب حياخذ الفورين كيي حق القسم بناء على الي هو بداخله مثلا لو القسم الي اسمه لست ون حقه الي دي 10 نلاحظ انه الطلاب اس ون واس تو الفورين كيي حق الاقسام حقهم بيكون هو 10
فهمت هه

الان بنسوي تحديث للبيانات من دون مانستخدم دالة الفايند عندك هذا الكود مثلا

```
var std = new Student
{
  Id = 11,
  Name = "S2.2",
  Email = "S2@s.com",
  Age = 11 ,
  Grade = 11,
  Birthdate = DateTime.Now,
```

```
  departmentId = 1
},

db.Update(std);
db.SaveChanges();
```

هنا هو تلقائيا بيروح بيدور على الطالب حسب الاي دي اعلاه وبيقوم بتعديل بياناته

وفيه طريقة ثانية بدل مانستخدم دالة التحديث هذه

```
db.Update(std);
```

نستخدم بدلها هذه الدالة

```
db.Entry(db.Student.Find(11)).CurrentValues.SetValues(std);
```

وهي استخدامها نفس السابقة ماشي اختلاف لاكن هنا فيه مشكلة في الدالتين وهو افترض انا عندي بعض الاعمدة تقبل نل وفي حالتها القديمة هي فيها قيم ولاكن انا م بغيت هذه القيم تتغير تجلس زي ماهي
بس بغيت اغير كم عمود بس هنا يقولك باستخدام هذه الدالتين لو م كتبت هذه الاعمدة داخل الاوبجيكت اعلاه هو تلقائيا رح يحط لها قيم نل ف الحل هو نكتب هكذا

```
var std = new Student
{
  Id = 11,
  Name = "S2.2",
  Email = "S2@s.com",
  Age = 11 ,
  Birthdate = DateTime.Now,
  departmentId = 1
},

db.Update(std);
db.Entry(db.Student.Find(11)).Property(x => x.Grade).IsModified = false;
db.SaveChanges();
```

نلاحظ اعلاه نقوله انه قيمة القراد لا تسوي لها تعديل خلها بقيمتها السابقة

طيبب الان لو نبغى تعديل جماعي بين بينات جدولين بينهم علاقة مثل هذا الكود

```
var db = new AppDbContext();

var departments = new List<Department>() {
   new Department()
{
```

```
   Id =  6 ,Name = "list 1", des = "xyz" ,
   students = new List<Student>() {
      new Student() {Id = 11, Name = "S1.1", Email = "S@s.com", Age = 10 , Grade = 10,
Birthdate = DateTime.Now },
      new Student() {Id = 12, Name = "S2.1", Email = "S2@s.com", Age = 11 , Grade = 11,
Birthdate = DateTime.Now },
   }
},
       new Department()
{
    Id = 7, Name = "list 2", des = "qaz" ,
   students = new List<Student>() {
      new Student() {Id = 10, Name = "S1.2", Email = "S@s.com", Age = 10 , Grade = 10,
Birthdate = DateTime.Now },
      new Student() {Id = 9, Name = "S2.2", Email = "S2@s.com", Age = 11 , Grade = 11,
Birthdate = DateTime.Now },
   }
},

   };


db.Departments.UpdateRange(departments);
db.SaveChanges();
```

في الكود اعلاه رح اعلاه يغير قيم الاقسام وبالمقابل رح يغير قيم الطلاب الي لهم علاقة بالاقسام اعلاه

طيب باقي معنا الحذف وهو سهل بس نستخدم دالة الريموف بس شوف الكود ادناه

```
var departments = new List<Department>() {
   new Department()
   {
      Id = 8, Name = "final dep 1", des = "xyz" ,
      students = new List<Student>() {
         new Student() {Id = 12,  Name = "final S1", Email = "final1@s.com", Age = 10 , Grade =
10, Birthdate = DateTime.Now },
         new Student() {Id = 14, Name = "final new S3", Email = "final3@s.com", Age = 12 ,
Grade = 12, Birthdate = DateTime.Now },
         new Student() {Name = "final S4", Email = "final4@s.com", Age = 12 , Grade = 12,
Birthdate = DateTime.Now },
      }
   }
};
```

```
db.Departments.UpdateRange(departments);
db.SaveChanges();
```

من قوة الاينتيتي فريموورك انه في المثال السابق هو بيروح بيعمل تحديث لجميع قيم الطلاب والاقسام ولاحظ انه في اخر اوبجيكت من كلاس الطلاب هو مامرر له الاي دي فهو تلقائيا بيفهم انه هذا صف جديد
فبيروح بيضيفة في قاعدة البيانات اما الاخرين بيعدلهم

الان معنا موضوع مهم وحلوو جدا وهو يتعلق بالأداء وسرعة التنفيذ بيقولك انك لما تجلب البيانات من قاعدة البيانات سواء كان للطباعة او للتعديل هو تلقائيا يعمل لها تراكير اي تتبع لحالة الجداول
طيب في بعض الحالات انك عندي كويري او لوجيك عشان اعرض البيانات ومش محتاج اعمل لها تتبع لاني م بغير فيها مثلا بس انا بغيتها للعرض في هذه الحالة وغيرها يفضل
انني اجلب البيانات بدوون عمل تتبع لها عشان يزيد الاداء خاصة مع البيانات الكبيرة طيب كيف نسوي هذا اول لنفترض اننا معي سطر بستخدم فيه
فنكشن لينكيو عشان تعمل كويري معينه وما ابغى اسوي لها تتبع اسوي هكذا

```
var dep = db.Departments.AsNoTracking().FirstOrDefault(x => x.id == 2);
dep.Name = "jfjjfjf";

db.SaveChanges();
```

نلاحظ ان الكود اعلاه ماراح يغير شي ابدا في قواعد البيانات لانه لن يتتبع التغيرات حتى مع اني قلت له قم بحفظ التغييرات لاكن ماشي بيتغير

طيب لو عندي كثير اسطر وكثير دوال في واجهه معينه وبغيت بسويلهم كلهم عدم تتبع مره وحده بدل م اكتب بعد كل فنكشن لينكيو عدم تتبع
نكتب هذا الكود

```
var db = new AppDbContext();

db.ChangeTracker.QueryTrackingBehavior = QueryTrackingBehavior.NoTracking;
```

بعد هذا السطر كل عملية على هذه الدي بي كونتيكتس رح تتم بدون عملية تتبع

طيب لو انا مثلا بغيت بعرف جميع الجداول الي عملها تتبع مع حالة تتبعها وهل تغيرت حالتها نعمل هذا الكود

```
var track = db.ChangeTracker.Entries();

forech(var tr in track)
{
  console.wrilteLine($"{tr.Entity.ToString()} ---->{tr.State}");
}
```

هذا الكود رح يعرض كل الجداول الي تم عمل لها تتبع مع حاالتغلتها هل تغيرت او لا

طيب لو انا مثلا عادي عندي يتم تتبع التغيرات لاكن عند نقطة معينه ابغى اوقف تتبع التغييرات اسوي هذا الكود

```
forech(var tr in track)
{
  console.wrilteLine($"{tr.Entity.ToString()} ---->{tr.State}");
  tr.State = EntityState.Detached;
}
```

هذا الكود بيلف على كل جدول من الي تم عمل لهم تغيرات وبيوقف التتبع عليهم

طيب الان تخيل معي كلاس خاص بالطلاب وفيه هذه الاعمدة زي الكود ادناه

```
public class Student
   {
      [Key]
      public int Id { get; set; }
      public string Name { get; set; }
      public string Email { get; set; }
      public int Age { get; set; }
      public int Grade { get; set; }
      public DateTime Birthdate { get; set; }
      public Grade grade { get; set; }

      [ForeignKey("department")]
      public int departmentId { get; set; }
      public Department department { get; set; }

      public ICollection<StudentBook> books { get; set; }

      //[NotMapped]
      //public ICollection<Attendance> attendances { get; set; }
   }
```

طيب لوقلت لك ابغى اوصل من خلال هذا الكلاس الى اسم القسم الي فيه الطالب مع اسم جميع الكتب الخاصه بالطالب
نسوي هذا الكود بعدين بشرحه

```
var db = new AppDbContext();
```

```
var stu = db.Students.Include(d => d.department)
                .Include(g => g.grade)
                .Include(b => b.books)
                .ThenInclude(sb => sb.book)
                .SingleOrDefault(s => s.Id == 3);

Console.WriteLine(stu.department.Name);

foreach (var book in stu.books)
{
   Console.WriteLine(book.book.Name);
}
```

طيب لما ابغى اوصل لبيانات جدول له علاقة بجدول معين من خلال الجدول الاول نستخدم دالة انكلود لاكن لما تكون العلاقة متعدد الى متعدد بيكون اكيد فيه جدول وسيط عندك مثلا فوق في المثال
عملت انكلود للجدول الوسيط بين الطلاب والكتب و هذا الجدول الوسيط يحتوي بس على اي دي الطالب مع اي دي الكتاب فعشان اعبر عبر هذا الجدول الوسيط لجدول الكتب نسوي دالة ذين انكلود زي ماهو اعلاه
لاكن هذه الطريقة تسمى ليجر لودينج اي معناها التحميل المبكر وعيوبها تقوم بجلب جميع البيانات من قاعدة البيانات لكل الجدوال

طيب زي ماقلنا ان التحميل مبكر عيبه انه يجي بالداتا كلها مع انه بحتاجها مش بالبداية
تخيل لو انه انا احتاج اول حاجه بغيت بجيب بيانات طالب عشان اسوي عليها عمليات في المرحلة الحالية انا مش محتاج مثلا لبيانات الاقسام او الكتب فمش معقول انني من الان بعمل انكلود وبجيب الداتا كلها من البداية
لا انا بجيب بيانات الطالب ومتى م احتجت لبيانات الاقسام او الكتب بجيبهم عن طريق هذه الطريقة

```
var db = new AppDbContext();

var stu = db.Students.SingleOrDefault(s => s.Id == 3);



db.Entry(stu).Reference(d => d.department).Load();
Console.WriteLine(stu.department.Name);


//db.Entry(stu).Collection(st => st.books).Load();



foreach (var book in stu.books)
{
   db.Entry(book).Reference(b => b.book).Load();
   Console.WriteLine(book.book.Name);
}
```

نلاحظ انني اولا جبت بيانات الطالب وبعدين احتجت للاقسام وجبتهم بالطريقة الي فوق وبعد فترة احتجت لبيانات الكتب وبرضه استخدمت الطريقة حق الاقسام وجبت الكتب
وطبعا الكود الي داخل الفور ايتش هو لانه جدول الكتب وسيط واحتاج عن طريقة اوصل لاسامي الكتب

فلو تلاحظ انه في الكتب استخدمنا كوليكشن وفي الاقسام استخدمنا ريفيرنس طيب متى نستخدم ذا ومتى نستخدم ذاك يقولك لو انك تبغى توصل لبيانات جدول عن طريق نافيجاشن بروبيرتي مثل الاقسام
في هذه الحالة نستخدم ريفيرنس اما اذا كان النافيجاشن بروبيرتي من نوع كوليكشن مثل الكتب نستخدم كوليكشن

طيب هذه الطريقة تسمى ايكسبليكت لودينج ومعناها التحميل الصريح
طيب في المثال السابق هو بيجيب البيانات كاملة بدون فلترة طيب لو ابغى اجيبها بفلترة بس نمسح كلمة لوود ونكتب بدلها هذا الكود ادناه

```
db.Entry(stu).Collection(st => st.books).Query().Where(x => x.bookId == 3).ToList();
```

طيب ي طويل العمر يقولك في حاجه توفر عليك كل الشغل الي فوق ومن خلالها تقدر تعمل لود للداتا او البيانات فقط بمجرد ما انك تكتب اسم الاوبجيكت وتعمل بعده دوت او نقطة
وبعدها تكتب اسم النافيجاشن بروبيرتي هو على طول بيعمل لود للداتا بلا وجع راس هذه الطريقة تسمى لازي لودينج ومعناها التحميل المتاخر وعشان نستخدمها لازم ننزل مكتبة اسمها هي

```
microsoft.entityframeworkcore.Proxes
```

وبرضة عشان نستخدمها لازم نروح نكتب قدام كل نافيجاشن بروبيرتي نكتب كلمة فيرشوال زي كذا

```
public virtual Department department { get; set; }
```

و برضه عشان نستخدمها نروح للدي بي كونتيكتس كلاس ونروح للدالة الكونفيورينج ونكتب قبل اليوز اس كيو ال سيرفير نكتب هذا الكود ادناه

```
protected override void OnConfiguring(DbContextOptionsBuilder options)
    {
       options.UseLazyLoadingProxies().UseSqlServer(Connections.sqlConStr);
    }
```

فبيكون شكل الكود الي جبنا فيه الداتا باستخدام طريقة التحميل الصريح بيكون شكلة هكذا

```
var db = new AppDbContext();

var stu = db.Students.SingleOrDefault(s => s.Id == 3);

Console.WriteLine(stu.department.Name);

foreach (var book in stu.books)
```

```
{
   Console.WriteLine(book.book.Name);
}
```

الان معنا ي حبيب القلب بيقولك كيف تعمل ترانز اكشن زي لما كنا نعمل في قواعد البيانات زي هكذا الكود

```
var db = new AppDbContext();

using var transaction = db.Database.BeginTransaction();

try
{
   db.Departments.Add(new Department { Name = "tarna 11", des = "edc" });
   db.SaveChanges();

   db.Departments.Add(new Department { Name = "tarna 44", des = "edc4" });
   db.SaveChanges();

   transaction.CreateSavepoint("data_ok");

   db.Departments.Add(new Department { Name = "tarna 33", des = "edc1" });
   db.SaveChanges();

   db.Departments.Add(new Department { Name = "tarna 22", des = "edcjjghghghg" });
   db.SaveChanges();


   transaction.Commit();
}
catch (Exception ex)
{
   Console.WriteLine(ex.Message);
   //transaction.Rollback();
   try
   {
      transaction.RollbackToSavepoint("data_ok");
      transaction.Commit();
   }
   catch (Exception l2ex)
   {
      Console.WriteLine(l2ex.Message);
      transaction.Rollback();
```

```
    }
}
```

طبعا هنا استخدمنا امر الساف بوينت عشان لو اول امرين تنفذوا بشكل صحيح وحصل خطا في باقي الاوامر يروح ينفذ اول امرين ولا ينفذ باقي الاوامر

وبكذا نكووون خلصنا تلخيص الكوورس بحمد الله عزوجل وتوووفيقة الحمدالله دائما وأبداا