LINQ (LANGUAGE INTEGRATED QUERY)

"Dil ile Bütünleşik Sorgu" anlamına gelen, .NET dillerinde veritabanı sorgulama işlemini kolayca gerçekleştirebilmemizi sağlayan, Microsoft .NET Framework bileşenidir.

Standart bir Linq yapısı şu şekildedir.

var DEGİŞKENADI **= from** Nesne **in** NesneKoleksiyonuveyaDizi

where ŞartOlacaksa

select SeçilecekAlanlar(Nesne)

LINQ sorgularında geri dönen sonuç genellikle var tipinden değişkenlere aktarılır.

"in" anahtar sözcüğünden sonra Koleksiyon veya Dizi verilir. İşte bu koleksiyon veya dizinin içinde dönüldüğünde, elde edilen her bir eleman **"in"** anahtar sözcüğünden önceki, ya da **"from"** anahtar sözcüğünden hemen sonraki **"Nesne"** diye adlandırılan değişkende tutulur.

Varsa şartımız **"where"** anahtar sözcüğünden sonra bu değişken üzerinden yapılır.

Ve en son **"select"** deyimiyle bu değişkenimiz **"Nesne"** arka planda oluşuturulan bir koleksiyona atılıp, var tipinden olan **"Değişkenadı"** adındaki değişkene bu koleksiyon gönderilir.

Bir örnekle açıklamak gerekirse;

```
sayılar dizisinde 10dan büyük sayıları bulmak istersek
int[] Sayilar = new int[]{1,3,5,7,8,9,18,22,34,59};
for (int i = 0; i < Sayilar.Length; i++)
  {
     if (Sayilar[i] > 10)
     {
        listBox1.Items.Add(Sayilar[i]);
     }
  }
```

```
Aynı işlemi Linq kullanarak yapmak istersek
var istenensonuc = from eleman in Sayilar
                   where eleman > 10
                   select eleman;
  foreach (var bulunan in istenensonuc)
  {
     listBox1.Items.Add(bulunan);
  }
```

Anonymous Type Kullanmak

LINQ sorgulamalarında sonuç kümelerini ya var tipinden değişken ile ya da tipinden emin olduğumuz koleksiyona veya diziye çevirerek sonuç döndürülebiliyordu. Ya da kendi belirlediğimiz bir tip geri döndürüyorduk. LINQ sorgusu sonucunda elimize geçen sonuç kümesindeki nesnelerin ya da değerlerin, bazı özelliklerini ya da tüm değerlerini, isimsiz bir nesne üzerinden geri dönmeye **Anonymous Type** deniyor.

Örneğin; bir veri tabanından veri çekerken, o tablodaki tüm alanların verilerini çekmek yerine, sadece belirli alanların verilerini **istersek farklı bir isimle dahi** çekmemize imkan tanıyan bir esneklik sunuyor **Anonymous Type.**

Linq Çeşitleri

LINQ teknolojisi tamamen veri erişim işlemini daha kolay ve anlaşılır hale getirmek amaçlı tasarlanmıştır. Bu veri erişim yöntemleri LINQ TO XML, LINQ TO OBJECT ve LINQ TO SQL olarak veri kaynağına göre değişiklik göstermektedir. Biz Linq To Sql yöntemini kullanıcaz.

Linq To Sql, sadece sorgunun şartına göre sonuç tabloları döndürmekten ibaret bir teknoloji değildir. Kayıt ekleyebilir, güncelleyebilir ve silebilirsiniz. Ve bu yaptığınız işlemler veritabanına da yansıyacaktır.

Tüm bu söylediklerimizi bir örnek üzerinde uygulayalım.

Örnekleri uygulamak için aşağıda alan bilgileri ve örnek kayıtları verilen öğrenci bilgilerini tutan **Ogrenciler,** ders bilgilerini tutan **Dersler** ve not bilgilerini tutan **notlar** isimli birbiri ile ilişkili üç tablo kullanılacaktır.

Dersler – Ogrenciler – Notlar Tabloları kayıt alanları

	Name	Data Type	Allow Nulls
🔑	DersID	int	☐
	DersAdi	nvarchar(50)	☑
			☐

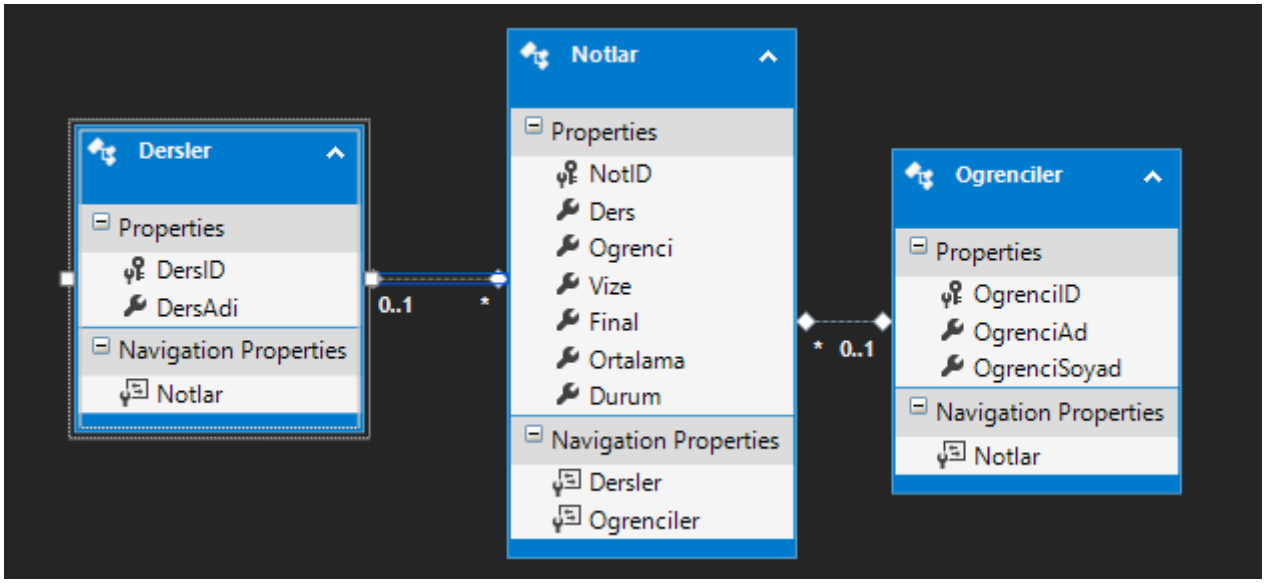
	Name	Data Type	Allow Nulls	Default	Length
🔑	NotID	int	☐		
	Ders	int	☑		
	Ogrenci	int	☑		
	Vize	int	☑		
	Final	int	☑		
	Ortalama	decimal(5,2)	☑		
	Durum	bit	☑		
			☐		

	Name	Data Type	Allow Nulls	Default	Length
🔑	OgrenciID	int	☐		
	OgrenciAd	nvarchar(50)	☑		50
	OgrenciSoyad	nvarchar(50)	☑		50
			☐		

Tablolar Arasındaki İlişki

Dersler tablosundaki **DersID** alanı ile Notlar tablosundaki **Ders** alanı arasında ikincil anahtar ilişkisi vardır.

Ogrenciler tablosundaki **OgrenciID** alanı ile Notlar tablosundaki **Ogrenci** alanı arasında ikincil anahtar ilişkisi vardır.



Örnek Kayıtlar

Tablolar üzerinde işlemlerin daha rahat yapılabilmesi için aşağıdaki örnek kayıtlar girilmiştir.

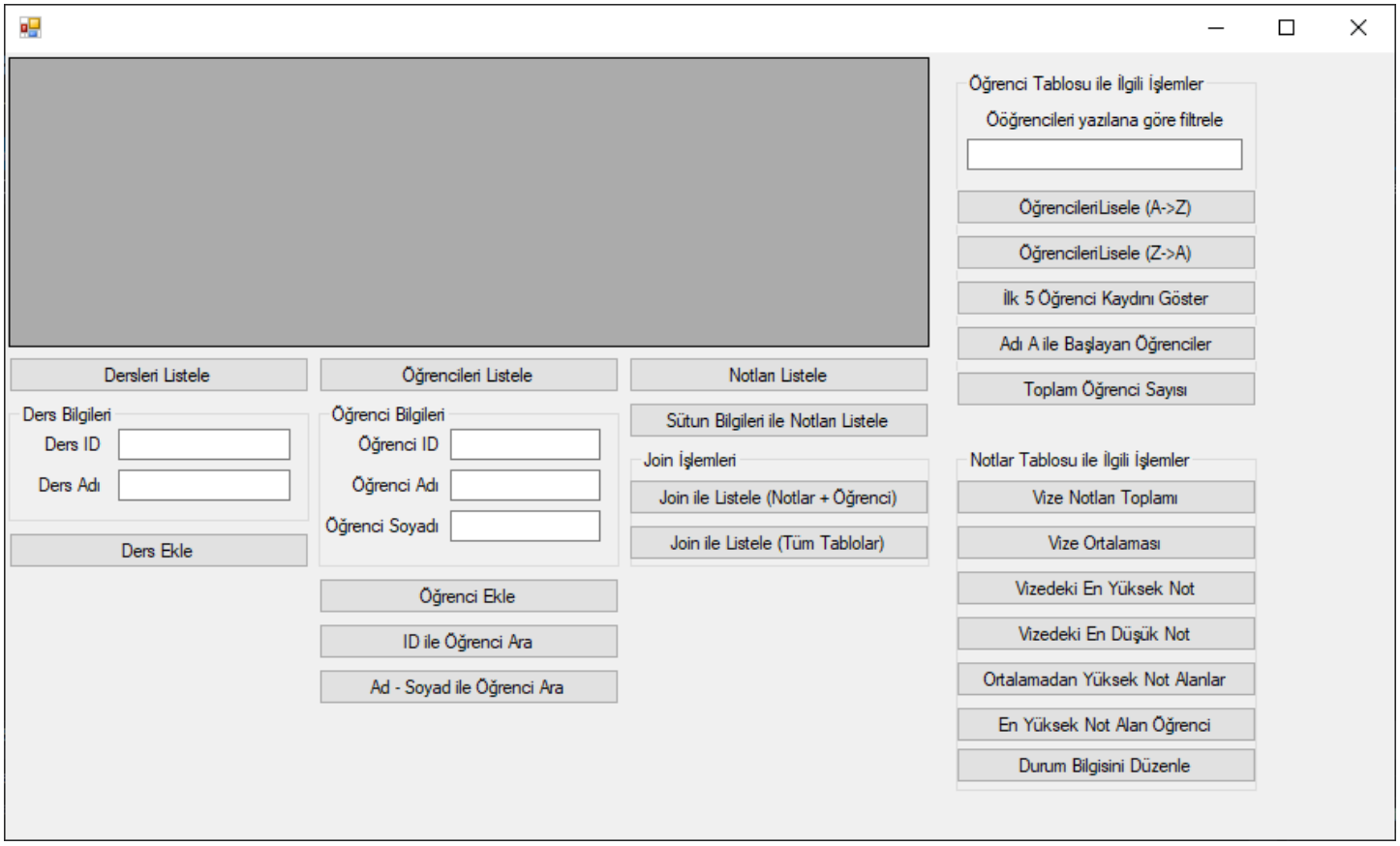
DersID	DersAdi
1	Fizik
2	Kimya
3	Biyoloji
4	Matematik
5	Edebiyat
6	Tarih
7	Coğrafya
NULL	*NULL*

OgrenciID	OgrenciAd	OgrenciSoyad
1	Bekir	Kurt
2	Muhsin	Karataş
3	Volkan	Tuna
4	Ayşe	Karataş
5	Canan	Kurt
6	Volkan	Keser
7	Ayşe	Çakır
NULL	*NULL*	*NULL*

NotID	Ders	Ogrenci	Vize	Final	Ortalama	Durum
1	1	1	40	50	45,00	False
2	1	2	50	60	55,00	True
3	2	2	50	60	55,00	True
4	2	3	30	40	35,00	False
5	2	4	50	40	45,00	False
6	3	1	60	60	60,00	True
7	4	1	55	75	65,00	True
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Form Tasarımı

Bu dokümanda aşağıda resmi bulunan örnekler açıklanacaktır. Örnekler belirlenirken mümkün olduğunca benzer içerikteki çözümlerin tekrarından kaçınılmıştır.



Ders Listele:

Dersler tablosundaki tüm kayıtların gridview da görüntülenmesini sağlayan metottur. **db.Dersler.ToList()** alternatifi kullanıldığında tablodaki tüm kayıt alanları listelenir. Tüm alanların listelenmesini istemiyorsak **Anonymous** tip kullanarak listeyi özelleştirebiliriz.

```
/*
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = db.Dersler.ToList();
dataGridView1.DataSource = listelenecek.ToList();
*/
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = from kayit in db.Dersler
                   select new { kayit.DersID, kayit.DersAdi };
dataGridView1.DataSource = listelenecek.ToList();
```

Öğrenci Listele:

Öğrenciler tablosundaki tüm kayıtların gridview da görüntülenmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = from kayit in db.Ogrenciler
                   select new {kayit.OgrenciID,kayit.OgrenciAd,kayit.OgrenciSoyad};
dataGridView1.DataSource = listelenecek.ToList();
```

Notları Listele:

Notlar tablosundaki tüm kayıtların gridview da görüntülenmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = from kayit in db.Notlar
                   select new { kayit.NotID, kayit.Ders,kayit.Ogrenci,
                                kayit.Vize,kayit.Final,kayit.Ortalama,kayit.Durum};
dataGridView1.DataSource = listelenecek.ToList();
```

Sütun Bilgileri ile Notları Listele:

Notlar tablosundaki tüm kayıtların gridview da görüntülenmesini sağlayan metottur. Bunu yaparken tabloda öğrenci ve ders alanlarını ilgili tablodan alarak id yerine isimlerinin listelenmesini sağlar.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = from kayit in db.Notlar
                   select new
                   {
                       kayit.NotID,
                       //kayit.Ders, Yerine
                       kayit.Dersler.DersAdi,
                       //kayit.Ogrenci, Yerine
                       kayit.Ogrenciler.OgrenciAd,
                       kayit.Ogrenciler.OgrenciSoyad,
                       kayit.Vize,
                       kayit.Final,
                       kayit.Ortalama,
                       kayit.Durum
                   };
dataGridView1.DataSource = listelenecek.ToList();
```

Ders Ekle:

Ders Adı isimli textbox a girilen bilginin **Dersler** tablosuna eklenmesini sağlayan metottur. **db.SaveChanges** komutunu kullanmadan yapılan değişiklik veritabanı üzerinde etkili olmaz.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
Dersler yeniders = new Dersler();
yeniders.DersAdi = txtDersAd.Text;
db.Dersler.Add(yeniders);
db.SaveChanges();
MessageBox.Show("Ders Listeye Eklendi");
```

Öğrenci Ekle:

Öğrenci Adı ve Öğrenci Soyadı isimli textboxlara girilen bilginin **Ogrenciler** tablosuna eklenmesini sağlayan metottur. **db.SaveChanges** komutunu kullanmadan yapılan değişiklik veritabanı üzerinde etkili olmaz.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
Ogrenciler yeniogrenci = new Ogrenciler();
yeniogrenci.OgrenciAd = txtOgrenciAd.Text;
yeniogrenci.OgrenciSoyad = txtOgrenciSoyad.Text;
db.Ogrenciler.Add(yeniogrenci);
db.SaveChanges();
MessageBox.Show("Öğrenci Listeye Eklendi");
```

ID ile Öğrenci Ara:

ÖğrenciID isimli textbox a girilen bilginin **Ogrenciler** tablosunda yer alıp almadığını **Find** fonksiyonu yardımı ile kontrol eder.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
int aranan = Convert.ToInt32(txtOgrenciID.Text);
var bul = db.Ogrenciler.Find(aranan);
if (bul== null)
{
    MessageBox.Show("Böyle bir öğrenci yok");
}
else
{
    txtOgrenciID.Text = bul.OgrenciID.ToString();
    txtOgrenciAd.Text = bul.OgrenciAd.ToString();
    txtOgrenciSoyad.Text = bul.OgrenciSoyad.ToString();
    MessageBox.Show("Aranan Öğrenci Listelendi");
}
```

Ad Soyad ile Öğrenci Ara:

Öğrenci Adı ve Öğrenci Soyadı isimli textboxlara girilen bilginin **Ogrenciler** tablosunda yer alıp almadığını kontrol eder. Bulunan ilk kaydı **Öğrenci Bilgileri** alanındaki ilgili textbox ta gösterir.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
string arananAd = txtOgrenciAd.Text;
string arananSoyad = txtOgrenciSoyad.Text;
var sorgu = db.Ogrenciler.Where(x => x.OgrenciAd == arananAd && x.OgrenciSoyad == arananSoyad).FirstOrDefault();
txtOgrenciID.Text = sorgu.OgrenciID.ToString();
txtOgrenciAd.Text = sorgu.OgrenciAd.ToString();
txtOgrenciSoyad.Text = sorgu.OgrenciSoyad.ToString();
```

Öğrencileri Listele (A->Z):

Öğrenci tablosundaki bilgileri A’ dan Z’ ye sıralanmış şekilde listelenmesini sağlar.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = db.Ogrenciler.OrderBy(x => x.OgrenciAd).ToList();
dataGridView1.DataSource = listelenecek;
```

Öğrencileri Listele (Z->A):

Öğrenci tablosundaki bilgileri Z' den A' ya sıralanmış şekilde listelenmesini sağlar.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = db.Ogrenciler.OrderByDescending(x => x.OgrenciAd).ToList();
dataGridView1.DataSource = listelenecek;
```

Öğrencileri Yazılana Göre Filtrele:

Sadece Textbox a girilen bilgiyi içeren kayıtların listelenmesini sağlayan metottur. Bu işlemi yapabilmek için metot textbox ın **TextChanged** özelliği ile tetiklenmektedir.

```
private void txtContains_TextChanged(object sender, EventArgs e)
{
    CFLinqOrnekEntities db = new CFLinqOrnekEntities();
    string aranan = txtContains.Text;
    var gosterilecek = from bul in db.Ogrenciler
                       where (bul.OgrenciAd.Contains(aranan))
                       select bul;
    dataGridView1.DataSource = gosterilecek.ToList();
}
```

Sadece İlk 5 Kaydı Göster:

Öğrenci tablosundaki kayıtların ilk 5 tanesinin listelenmesini sağlar.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = db.Ogrenciler.Take(5).ToList();
dataGridView1.DataSource = listelenecek;
```

Adı "A" ile Başlayan Öğrencileri Göster:

Öğrenci tablosundaki kayıtlardan OgrenciAd bilgisi "A" ile başlayan kayıtların listelenmesini sağlar.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var listelenecek = db.Ogrenciler.Where(x => x.OgrenciAd.StartsWith("a")).ToList();
dataGridView1.DataSource = listelenecek;
```

Toplam Öğrenci Sayısını Göster:

Öğrenciler tablosundaki kayıt sayısını MessageBox ta gösterilmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
int toplamogrenci = db.Ogrenciler.Count();
MessageBox.Show(toplamogrenci.ToString(), "Toplam Öğrenci Sayısı");
```

Vize Notları Toplamı:

Notlar tablosunda yer alan kayıtların Vizeden aldıkları puanların toplamının MessageBox ta gösterilmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var vizetoplam = db.Notlar.Sum(x => x.Vize);
MessageBox.Show(vizetoplam.ToString(), "Vize Notları Toplamı");
```

Vize Ortalaması:

Notlar tablosunda yer alan kayıtların Vizeden aldıkları puanların ortalamasını MessageBox ta gösterilmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var vizeortalama = db.Notlar.Average(x => x.Vize);
MessageBox.Show(vizeortalama.ToString(), "Vize Notları Ortalaması");
```

Vizedeki En Yüksek Not:

Notlar tablosunda yer alan kayıtların Vizeden aldıkları puanların en yükseğini MessageBox ta gösterilmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var vizeMax = db.Notlar.Max(x => x.Vize);
MessageBox.Show(vizeMax.ToString(), "Vize Notların En Yükseği");
```

Vizedeki En Düşük Not:

Notlar tablosunda yer alan kayıtların Vizeden aldıkları puanların en düşüğünü MessageBox ta gösterilmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var vizeMin = db.Notlar.Min(x => x.Vize);
MessageBox.Show(vizeMin.ToString(), "Vize Notların En Düşüğü");
```

Ortalamadan Yüksek Alanlar:

Notlar tablosunda yer alan kayıtların Vizeden aldıkları puanları, vize ortalamasından yüksek olan kayıtların listelenmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var vizeortalama = db.Notlar.Average(x => x.Vize);
var listelenecek = db.Notlar.Where(x => x.Vize >= vizeortalama).ToList();
dataGridView1.DataSource = listelenecek;
```

En Yüksek Not Alan Öğrenci:

Notlar tablosunda yer alan kayıtların Vizede en yüksek puan alan öğrencinin listelenmesini sağlayan metottur.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
//En Yüksek vize notu bulundu
var enyuksekvize = db.Notlar.Max(x => x.Vize);
//En yüksek vize notuna sahip öğrenci bulundu
var bulunan = db.Notlar.Where(x => x.Vize == enyuksekvize).FirstOrDefault();
//Öğrenci ID bulundu
int  aranan = Convert.ToInt32(bulunan.Ogrenci.ToString());
//Öğrenci bilgileri gösterildi
var listelenecek = db.Ogrenciler.Where(y => y.OgrenciID == aranan);
dataGridView1.DataSource = listelenecek.ToList();
```

Durum Bilgilerini Güncelle:

Notlar tablosunda yer alan kayıtların **ortalama**larına göre, notu 50 den az olanları FALSE, 50' den fazla olanları TRUE olarak değiştirir.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var sorgu = db.Notlar.ToList();
foreach(var guncelle in sorgu)
{
    if (guncelle.Ortalama >= 50)
    {
        guncelle.Durum = true;
    }
    else
    {
        guncelle.Durum = false;
    }
}
db.SaveChanges();
dataGridView1.DataSource = sorgu.ToList();
```

Join İle Listele (Notlar + Öğrenci):

Join kullanarak birden fazla tablodan alınan verilerle listelenen içeriği daha düzenli hale getirebiliriz.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var sorgu = from d1 in db.Notlar
            join d2 in db.Ogrenciler
            on d1.Ogrenci equals d2.OgrenciID
            select new
            {
                ÖĞRENCİ = d2.OgrenciAd + " " + d2.OgrenciSoyad,
                VİZE = d1.Vize,
                FİNAL = d1.Final
            };
dataGridView1.DataSource = sorgu.ToList();
```

Join İle Listele (Tüm Tablolar):

Join kullanarak birden fazla tablodan alınan verilerle listelenen içeriği daha düzenli hale getirebiliriz.

```
CFLinqOrnekEntities db = new CFLinqOrnekEntities();
var sorgu = from d1 in db.Notlar
            join d2 in db.Ogrenciler
            on d1.Ogrenci equals d2.OgrenciID
            join d3 in db.Dersler
            on d1.Ders equals d3.DersID
            select new
            {
                ÖĞRENCİ = d2.OgrenciAd + " " + d2.OgrenciSoyad,
                DERS = d3.DersAdi,
                VİZE = d1.Vize,
                FİNAL = d1.Final,
                //Eğer durum = true ise GEÇTİ, değilse KALDI yaz
                DURUMU = d1.Durum == true?"GEÇTİ":"KALDI"
            };
dataGridView1.DataSource = sorgu.ToList();
```