Bu projenin daha detaylı kısmı github repsunda mevcur link aşağda veriyorum
Link: https://github.com/mehmet-karataslar/SinavOgrenciSistemi

---

1) Bu proje gerçekte ne yapıyor?

SinavOgrenciSistemi, bir okul/kurum senaryosunda:

- **Öğrencileri** yönetir (ekle–sil–güncelle–listele),
- Öğrencilerin bağlı olduğu **kulüpleri** yönetir (öğrenci bir kulübe bağlı olabilir, olmayabilir),
- **Dersleri** yönetir,
- Her öğrenci için ders bazında **sınav notlarını** saklar (Sınav1/2/3),
- Notlardan **ortalama** ve **geçti/kaldı durumunu** üretir,
- Bunları hem **Entity Framework (LINQ)** ile hem de **SQL'in "kurumsal araçları" (Stored Procedure, Function, Transaction)** ile yapar.

Yani proje bir "demo" değil; **EF'in günlük kullanımını** (CRUD + LINQ + ilişkiler) ve **gerçek hayatta veritabanında yapılan ileri işleri** (SP/Function/Transaction) tek bir uygulamada gösteren bir eğitim projesi.

---

2) Teknoloji seçimi ve ne anlama geliyor?

- **.NET Framework 4.8 + WinForms**: Masaüstü, hızlı UI geliştirme. Form tabanlı akış var.
- **Entity Framework 6.4.4**: ORM. Veritabanındaki tabloları C# sınıfları gibi kullanıyorsun.
- **SQL Server**: Projenin omurgası. İleri işlerin (SP/Function/Transaction) hepsi burada.

Bu kombinasyonun en büyük artısı:
C# tarafında hızlı geliştirme + SQL tarafında güçlü raporlama ve güvenli işlem yönetimi.

---

3) Sistem mimarisi: "Ekranlar neyi temsil ediyor?"

Bu projede ekranlar (formlar) "rastgele yapılmış sayfalar" değil; her biri Entity Framework'ün farklı bir yeteneğini göstermek için ayrılmış **öğrenme modülleri** gibi çalışıyor:

FormMain (Ana Menü)

- Tek işi: kullanıcıyı doğru modüle yönlendirmek.
- 7 buton → 7 eğitim modülü.
- Bu yaklaşım iyi: "her şey tek ekranda karmaşa" yerine, yetenekleri bölerek gösteriyorsun.

---

4) Veri modeli: 4 tabloyla kurulan gerçek ilişki

Veritabanında 4 ana tablo var:

1) TBLOGRENCI (Öğrenci)

- Kimlik, ad, soyad, fotoğraf yolu, kulüp id.
- **Kulüp opsiyonel**: yani öğrenci kulüpsüz olabilir. Bu, gerçek hayata uygun.

2) TBLKULUPLER (Kulüp)

- Kulüp id, kulüp adı.
- Bir kulübün birden çok öğrencisi olabilir (**one-to-many**).

3) TBLDERSLER (Ders)

- Ders id, ders adı.
- Bir dersin birden çok not kaydı olabilir (**one-to-many**).

4) TBLNOTLAR (Notlar)

- Asıl "iş" burada: ders + öğrenci + sınavlar + ortalama + durum.
- Öğrenci ve ders id'si zorunlu: "not kaydı sahipsiz olamaz".
- Ortalama ve durum çoğunlukla "hesaplanmış veri" gibi davranıyor (bazı işlemlerde SP hesaplıyor).

Modelin kritik noktası:
Notlar tablosu, "Öğrenci" ile "Ders" arasında bir **bağlantı/ilişki tablosu** gibi davranıyor. Çünkü her not kaydı hem bir öğrenciye hem bir derse bağlı.

---

5) Entity Framework tarafı: "Navigation Property neden bu kadar önemli?"

Bu projede EF tarafının merkezinde şu fikir var:

"JOIN yazma, ilişkiler üzerinden yürü."

Örneğin notlar listesini çekerken:

- `n.TBLOGRENCI.OgrenciAd`

- `n.TBLDERSLER.DERSAD`

Bu sayede:

- SQL JOIN'i sen yazmıyorsun,
- EF ilişkiden join'i çıkarıyor,
- Kod okunabilirliği artıyor.

Bu proje bunu özellikle "Navigation" formunda bilinçli şekilde gösteriyor:
Notlar → öğrenci adı, **notlar → ders adı**, **3 tablo birleşimi** vs.

---

6) DTO'lar: "Neden entity değil de DTO?"

Projede SP/Function sonuçları için **DTO sınıfları** var. Bunun iki nedeni var:

1. SP/Function çıktısı her zaman birebir tablo yapısı değildir.
   Mesela "ders başarı raporu" bir tablo satırı değil, "toplam öğrenci, geçen, kalan, başarı yüzdesi" gibi bir rapor çıktısıdır.
2. Entity sınıfları veritabanı şemasına bağlıdır.
   Rapor çıktıları ise "sunum amaçlı"dır.

Bu yüzden:

- `OgrenciDetay`, `DersBasariRapor`, `NotFiltreSonuc` gibi DTO'lar,
- **strongly-typed** veri taşımayı sağlar,
- SP/Function sonuçlarını C# tarafında temiz kullanmanı sağlar.

Bu yaklaşım kurumsalda standarttır.

---

7) Formlar ve işlevsel akış: "Her form neyi kanıtlıyor?"

A) FormTemelEF (Soru 1–5)

Amaç: EF'in en temel kullanımını göstermek.

- DbContext'in form seviyesinde yaşaması
- Listeleme
- Alan gizleme (fotoğrafı göstermeme)
- EF vs ADO.NET karşılaştırması

Buradaki mesaj şu:
EF hızlı, okunur, tip güvenli. ADO.NET daha "ham" ama bazen gerekli.

B) FormNavigation (Soru 6–12)

Amaç: ilişkilerin gücü.

- Notları listelerken öğrenci/ders adını navigation ile almak
- Ad+soyad birleştirme
- 3 tabloyu birlikte listeleme
- Bir öğrencinin tüm notlarını "collection navigation" üzerinden alma

Burada proje şunu öğretiyor:
İlişkiyi doğru kurarsan sorgu basitleşir.

C) FormCRUD (Soru 13–18)

Amaç: uygulamanın gerçek omurgası.

- Ekleme (Create)
- Silme (Delete) — kritik detay: önce notlar siliniyor
- Güncelleme (Update)
- Grid'den seç → inputları doldur
- EntityState mantığı

Bu modül, EF'in "tracking" sistemini gösteriyor:
Entity'nin `Added/Modified/Deleted` olma mantığı.

D) FormLINQ (Soru 19–25)

Amaç: kullanıcı davranışı ile sorgu yazımı.

- Canlı arama (TextChanged)
- Sıralama
- İlk 3 kayıt
- StartsWith/EndsWith
- ID ile filtre

Bu form şunu kanıtlıyor:
UI olayları → LINQ sorgusu → DataGridView döngüsü çok net.

E) FormAggregate (Soru 26–28)

Amaç: raporlama mantığı.

- Sum

- Average
- Ortalama üstü filtreleme

Önemli pratik: null notlar varsa `?? 0` veya filtreleme şart.

F) FormDurum (Soru 29–30)

Amaç: “durum” konsepti.

- Durum listesi (Geçti/Kaldı)
- Geçenleri ve kalanları iki ayrı gridde göstermek

Bu, UX açısından da güzel: tek listede boğmak yerine ayrıştırma.

G) FormGelismis (Soru 30–45)

Amaç: “kurumsal SQL özellikleri + EF entegrasyonu”.
3 sekme:

1) Stored Procedures

- Kontrollü not ekleme (duplicate kontrolü)
- Öğrenci detay raporu
- Ders başarı raporu (MessageBox rapor)
- Not güncelleme (ortalama+dürüm SP’de hesaplanıyor)
- Toplu silme (silinen kayıt sayısı dönüyor)

Buradaki mesaj:
Kurallar DB tarafında da yaşayabilir.

2) Functions

- Scalar function (tek değer döner)
- TVF (tablo döner)
- Çok parametreli filtreleme (opsiyonel parametre + DBNull)
- Kulüp başarı dashboard
- Zayıf öğrenciler listesi
- Öğrenci not özeti (label’lara yazma)

Buradaki mesaj:
Raporlama ve analitik sorgular için Function iyi bir araçtır.

3) Transactions

- Çok adımlı ekleme (öğrenci + 3 ders notu)
- TransactionScope ile toplu güncelleme

- Çok tablolu silme (atomic)
- Transaction içinde SP çağırma
- File.Exists doğrulamasıyla güvenli güncelleme

Buradaki mesaj:
Bir iş parçalıysa ve "ya hep ya hiç" olmalıysa transaction şarttır.

---

8) Projenin güçlü tarafları (gerçekçi değerlendirme)

Güçlü Yanlar

- **Modüler öğrenme tasarımı**: Her form belirli bir EF yeteneğini hedefliyor.
- **Hem EF hem SQL araçları var**: Tek taraflı değil.
- **DTO kullanımı doğru**: Rapor/çıktı işleri entity'den ayrılmış.
- **Transaction senaryoları gerçek hayatla uyumlu**: "çok adımlı iş" mantığı var.
- **UI akışı net**: Menu → modül → geri.

Zayıf/Riskli Yanlar (dürüst taraf)

- **Ortalama/Durum verisinin iki yerde hesaplanması riski**:
  Bazı yerlerde C# hesaplıyor, bazı yerlerde SP hesaplıyor. Bu tutarsızlık doğurabilir.
  En doğrusu: "tek otorite" belirlemek (ya DB hesaplar, ya servis katmanı).
- **Her formun kendi DbContext yönetmesi**: eğitim için kabul; gerçek projede merkezi servis/repository daha temiz olur.
- **WinForms veri bağlama katmanı**: büyürse test edilebilirlik düşer (UI'ye gömülen iş mantığı artar).
- **Lazy Loading** varsayılan açık**: büyük veri ve yanlış kullanımda performans sürprizi çıkarabilir.

Bunlar "ödev projesi" için normal ama "ürün" hedeflenseydi revize edilirdi.

---

9) Bu raporun tek cümlelik özeti

Bu proje, **WinForms üzerinde EF6 ile CRUD+LINQ+ilişkiler** dünyasını öğretirken, aynı anda **SQL Server'ın Stored Procedure/Function/Transaction** gücünü gösteren, **45 senaryoyu modüllere ayırarak sunan** kapsamlı bir masaüstü eğitim uygulamasıdır.