

# SQL Online Shopping Project

E-ticaret sitemin veritabanı tasarımı normalizasyonu göz önünde bulundurularak hazırlandı ve ilişkisel bir veritabanına dönüştürüldü. Öncelikle belirtilen ürünler, kategoriler, kullanıcılar ve siparişler tablolarıyla sınırlı kalmayarak, fazladan tablolar ekledim ve bunları birbirleriyle ilişkilendirdim. Final haftasında ve zaman kısıtlamaları nedeniyle tablo sayısını çok fazla artıramamış olsam da, tasarımı daha çeşitli ve etkili hale getirmek istedim.

Başlangıcımın ilk adımı, Excel üzerinde tabloları tasarlamak ve birbirleriyle ilişkilendirmek oldu. Bunu yapmamın sebebi, verileri ve tabloları görselleştirebilirim, sunumunun hem daha etkili olacağını hem de üzerinde çalışmamın daha rahat ve anlaşılır olacağını düşünmemdi. Tasarlarken renkler ile kolonların ilişkilerini belirttim.

Aşağıda Excel üzerinde yaptığım görselleştirmeyi paylaşıyorum:

	A	B	C	D	E	F
1	Urunler			Kategoriler		
2	UrunID	int	Primary,identity(1,1)	KategoriID	int	Primary,identity(1,1)
3	KategoriID	int	foreign	KategoriAdi	nvarchar(70)	
4	SaticilID	int	foreign			
5	UrunKodu	UNIQUEIDENTIFIER				
6	UrunAdi	nvarchar(70)				
7	UrunFiyat	int				
8	Stok	int				
9						
10						
11	OdemeTuru			Saticilar		
12	OdemeID	int	Primary,identity(1,1)	SaticilID	int	Primary,identity(1,1)
13	OdemeSekli	nvarchar(70)		SaticiAd	nvarchar(25)	
14				SaticiTelNo	nvarchar(25)	
15				SaticiEmail	nvarchar(50)	
16						
17						
18						

G	H	I	J	K	L
<b>Kullanıcılar</b>			<b>Siparisler</b>		
<b>UserID</b>	int	Primary,identity(1,1)	<b>SiparişID</b>	int	UNIQUEIDENTIFIER PRIMARY KEY
<b>AdresID</b>	int	foreign	<b>UserID</b>	int	foreign
<b>Ad</b>	nvarchar(25)		<b>UrunID</b>	int	foreign
<b>Soyad</b>	nvarchar(25)		<b>OdemeID</b>	int	foreign
<b>TelNo</b>	nvarchar(25)		<b>Adet</b>	int	
<b>Email</b>	nvarchar(50)		<b>SiparişTarihi</b>	date	
<b>KayıtTarihi</b>	date	trigger			
<b>GuncellemeTarihi</b>	date	trigger			
<b>Adresler</b>					
<b>AdresID</b>	int	Primary,identity(1,1)			
<b>AdresAdı</b>	nvarchar(25)				
<b>Adres</b>	nvarchar(300)				
<b>Şehir</b>	nvarchar(25)				

Taslakları hazırladıktan sonra sıra tabloları oluşturmaya geldi. Tüm kodları tek tek yazarak belgeyi uzun bir hale getirmek istemediğimden aşağıda örnek bir kod bırakıyorum:

```
CREATE TABLE Siparisler (
    SiparisID UNIQUEIDENTIFIER PRIMARY KEY,
    UserID INT FOREIGN KEY REFERENCES Kullanıcılar(UserID),
    UrunID INT FOREIGN KEY REFERENCES Urunler(UrunID),
    OdemeID INT FOREIGN KEY REFERENCES OdemeTuru(OdemeID),
    SiparisTarihi DATE
);
```

Tabloları oluşturduktan sonra verileri girmeden önce birkaç tetikleyici (trigger) eklemek istedim. Bu tetikleyicilerin veritabanımda otomasyon sağlamasını amaçladım. Eklediğim triggerlardan birkaçının amacı, veri girişi sırasında olası hatalardan biri olan tarih girişini otomatize etmektir. Öncelikle, kullanıcılar tabloma kullanıcı kaydı girdiğimde, kullanıcının kayıt tarihini GETDATE() komutu ile bu günün tarihi olarak çekmesini sağladım.

Ardından, kayıt tarihinde yaptığım uygulamayı kullanıcı bilgilerini güncellediğinde de güncelleme tarihi olarak yapmasını istedim ve bunun için de bir tetikleyici oluşturdum.

```

CREATE TRIGGER trg_AfterInsert_Kullanici
ON Kullanıcılar
AFTER INSERT
AS
BEGIN
    UPDATE K
    SET KayıtTarihi = GETDATE()
    FROM Kullanıcılar K
    INNER JOIN inserted I ON K.UserID = I.UserID;
END;

```

```

CREATE TRIGGER trg_AfterUpdate_Kullanıcılar
ON Kullanıcılar
AFTER UPDATE
AS
BEGIN
    UPDATE K
    SET GuncellemeTarihi = GETDATE()
    FROM Kullanıcılar K
    INNER JOIN inserted I ON K.UserID = I.UserID;
END;

```

Kullanıcılar tablosundaki tetikleyicilerimi bitirdikten sonra sipariş oluşturduğumuzda da güncel tarihi çekip giren bir trigger oluşturdum.

```

CREATE TRIGGER trg_AfterInsert_Siparisler
ON Siparisler
AFTER INSERT
AS
BEGIN
    UPDATE S
    SET SiparisTarihi = GETDATE()

```

```
FROM Siparisler S
INNER JOIN inserted I ON S.UserID = I.UserID;
END;
```

Ardından yine sipariş oluşturduğumuzda ürünler tablosuna ulaşp siparişte yazan sayı kadar stok miktarı azaltan ve stok bilgisinin yanlış olarak kalmasını engelleyen bir tetikleyici daha oluşturdum.

```
CREATE TRIGGER trg_StokMiktarı
ON Siparisler
AFTER INSERT
AS
BEGIN
    UPDATE Urunler
    SET Stok = Stok - S.Adet
    FROM Urunler u
    INNER JOIN inserted S ON u.UrunID = S.UrunID;
END;
```

Tetikleyicilerimi bitirdikten sonra artık verilerimi girmeye başladım. Verileri girerken gerçekçi olmalarına özen gösterdim. Birkaç ismi baz alarak, bu isimleri döngülerle belirli düzenlerde giriş yaptım. Bu işlemi sadece kullanıcılar tablosunda değil, aynı zamanda adresler, satıcılar ve ürünler tablolarımda da gerçekleştirdim. Kodlar uzun olduğu için kullanıcılar tablomun veri oluşturma kodunu aşağıya örnek olarak bırakıyorum.

```
begin transaction

declare @counter int
declare @ad nvarchar(max)
declare @soyad nvarchar(max)
```

```

declare @telno nvarchar(30);

set @counter = 0

while @counter < 100000
begin
    set @ad =(
        case
            when @counter % 9 = 0 then 'MEHMET'
            when @counter % 9 = 1 then 'HÜSEYİN'
            when @counter % 9 = 2 then 'YUSUF'
            when @counter % 9 = 3 then 'FATMA'
            when @counter % 9 = 4 then 'MERYEM'
            when @counter % 9 = 5 then 'ESRA'
            when @counter % 9 = 6 then 'OSMAN'
            when @counter % 9 = 7 then 'FATİH'
            when @counter % 9 = 8 then 'ÖZLEM'
            when @counter % 9 = 9 then 'KADİR'
        end
    );
    set @soyad =(
        case
            when @counter % 9 = 0 then 'VURAL'
            when @counter % 9 = 1 then 'KAYA'
            when @counter % 9 = 2 then 'ÖZER'
            when @counter % 9 = 3 then 'CANACANKATAN'
            when @counter % 9 = 4 then 'TEKİN'
            when @counter % 9 = 5 then 'KOÇ'
            when @counter % 9 = 6 then 'CÖMERT'
            when @counter % 9 = 7 then 'YÜKSEL'
            when @counter % 9 = 8 then 'KAYA'
            when @counter % 9 = 9 then 'ÇOLAK'
        end
    );
    set @telno =

```

```

        CONCAT(
            SUBSTRING(CAST(ROUND(RAND()), 10) * 1000000 AS NVARCHAR(10)),
            SUBSTRING(CAST(ROUND(RAND()), 10) * 100000 AS NVARCHAR(5))
        );

insert into Kullanıcılar(AdresID,Ad,Soyad,TelNo,Email)
values
(
    @counter,
    @ad,
    @soyad,
    @telno,
    'postadres'i' + CAST(@counter + 1 as nvarchar(100)) + '@gmail.com'
)
set @counter = @counter + 1
end
SELECT * FROM Kullanıcılar
commit

```

Tablolarımın verilerini girdikten sonra artık veritabanım sipariş oluşturmaya hazırды. Ancak her seferinde tek tek "INSERT INTO" ile siparişleri girmek istemedim. Bu sipariş giriř işini kolaylařtırmak için bir prosedür oluřturduđ. Bu prosedür sayesinde tablonun kolonları ve uzun kodlarla uğrařmama gerek kalmadan, kısa bir prosedür çağırma kodu ile verilerimi rahatlıkla girebildim.

```

CREATE PROCEDURE SiparisEkle
    @UserID INT,
    @UrunID INT,
    @OdemeID INT,
    @Adet INT
AS
BEGIN
    INSERT INTO Siparisler (SiparisID, UserID, UrunID, OdemeID, Adet)

```

```
VALUES (NEWID(), @UserID, @UrunID, @OdemeID, @Adet);
END;
--aynı şekilde prosedürü çağırma kodum
exec SiparisEkle
    @UserID = 1,
    @UrunID =282951,
    @OdemeID = 1,
    @Adet = 937
```

Siparişleri rahat bir şekilde girdikten sonra hangi kullanıcıların hangi adreslere sahip olduğunu, sipariş detaylarını ve hangi kullanıcının ne sipariş verdiğini tek tek uzun kodlar ile her seferinde kontrol etmek yerine, bu işlemi daha kullanışlı hale getirmek için view'lar oluşturdum. View'lerim üzerinden istediğim kolonları rahatça düzenleyebiliyor ve tek satırlık bir kod ile bu tabloyu ekrana getirebiliyorum.

- Kullanıcılarımın ne sipariş verdiğini gördüğüm view:

```
create view vw_Kullanıcı_Siparisleri as
Select K.Ad, K.Soyad, K.Email, S.SiparisID, S.Adet, S.SiparisTarihi
From Kullanıcılar K JOIN Siparisler S on K.UserID=S.UserID
```

- Kullanıcılarımın bilgilerini özetleyip hangi adrese ve iletişim adresine sahip olduğunu gösteren view:

```
create view vw_Kullanıcı_Adresleri as
Select K.Ad, K.Soyad, K.Email, K.TelNo, A.AdresAdı, A.Adres
From Kullanıcılar K JOIN Adresler A on K.AdresID=A.AdresID
```

- Siparişlerin tüm detaylarını ayrıntılı bir biçimde gösteren, toplam tutarı hesaplayan ve birçok tablonun birleşiminden oluşan view:

```
CREATE VIEW vw_SiparisDetaylarıToplamTutar AS
SELECT
```

```
s.SiparisID,  
k.Ad AS KullaniciAdi,  
k.Soyad AS KullaniciSoyadi,  
a.AdresAdi AS KullaniciAdresi,  
o.OdemeSekli AS OdemeTuru,  
u.UrunAdi,  
u.UrunFiyat,  
s.Adet,  
s.Adet * u.UrunFiyat AS ToplamTutar,  
s.SiparisTarihi  
FROM  
Siparisler s  
INNER JOIN Kullanicilar k ON s.UserID = k.UserID  
INNER JOIN Adresler a ON k.AdresID = a.AdresID  
INNER JOIN OdemeTuru o ON s.OdemeID = o.OdemeID  
INNER JOIN Urunler u ON s.UrunID = u.UrunID;
```

Öncelikle, bu eğitim fırsatını bize sağladığınız için başta Ahmet Kaya hocama, sonrasında da [www.techcareer.net](http://www.techcareer.net)'e çok teşekkür ederim. Bu bootcamp, sektöre dayalı çok verimli bilgiler öğrenmemize olanak sağladı ve SQL'in en ince ayrıntılarına kadar detaylı bir şekilde incelenmesini sağladı. Daha önce birkaç platform ve üniversite tarafından SQL dersi almış olmama rağmen, bu kadar derinlemesine öğrenmemiştim.

Bu süreçte özellikle veritabanı tasarımı, normalizasyon, trigger kullanımı gibi konular üzerinde detaylı çalışma imkanı buldum. Bunun yanı sıra, SQL'in performans optimizasyonu ve karmaşık sorguların yazımı gibi pratik becerileri de geliştirdim. Eğitim sürecinde elde ettiğim bu derin bilgileri, iş hayatımda etkili bir şekilde kullanmayı hedefliyorum.

Teşekkürler.

Musa Emir Doğan