BMÜ329 Veri Tabanı Sistemleri Dersi Dönem Projesi

ECZANE İLAÇ YÖNETİM SİSTEMİ

Proje Ekibindeki Kişiler

Öğrenci Adı	Öğrenci Numarası
Emirhan BAL	225260028
Mustafa TATLI	225260046
Ahmet Eren ARGUN	225260072

Bu proje, eczanelerin operasyonel süreçlerini dijitalleştirmek ve modernize etmek amacıyla geliştirilmiş bir Veri Tabanı Yönetim Sistemi projesidir. Proje, eczacılar ve müşteriler için tasarlanmış olup ilaç stok yönetimi, müşteri takibi, satış işlemleri ve rezervasyon süreçlerini optimize etmek için kapsamlı bir altyapı sunar.

Proje sayesinde:

•**Eczacılar**, ilaçların stok miktarlarını, tedarikçi bilgilerini ve son kullanma tarihlerini kolayca takip edebilir. Ayrıca, müşterilerin satın alma geçmişini görüntüleyerek kişiselleştirilmiş hizmet sağlayabilirler.
•**Müşteriler**, eczaneden almak istedikleri ilaçların stok durumunu görüntüleyebilir, ilaç fiyatlarını öğrenebilir ve rezervasyon işlemi yaparak stoktan ilaç ayırtabilir.
•**Şubeler**, şube bazlı stok takibi ve performans raporları oluşturarak yönetim süreçlerini daha verimli hale getirebilir.
•**Tedarikçiler**, ilaçların tedarik süreci ve ilişkili bilgilerle entegre şekilde yönetilebilir.

Bu proje, eczanelerin günlük operasyonlarında manuel iş yükünü azaltarak daha hızlı ve verimli bir yönetim sağlar. Modern veri tabanı tasarım prensiplerine uygun olarak geliştirilmiş ve SQL Server kullanılarak uygulanmıştır. Aynı zamanda, ilişkisel veri tabanı tasarımı BCNF ya da 3NF seviyesine normalleştirilmiş, kod bütünlüğü sağlanmış ve veri tabanı performansı optimize edilmiştir.

PROJE GEREKSİNİMLERİ

Müşteri Gereksinimleri

Gereksinim	Açıklama
İlaç Bilgilerini Görüntüleme	Müşteri, geçmişte aldığı ilaçların bilgilerini görüntüleyebilir.
Stok Kontrolü	Müşteri, almak istediği ilacın stok durumunu kontrol edebilir.
Rezervasyon yapma	Müşteri, istediği ilacı rezerve ederek stoktan ayırtabilir.
Fiyat Bilgisi Görüntüleme	Müşteri, ilaçların fiyatlarını görüntüleyebilir.

Eczacı Gereksinimleri

Gereksinim	Açıklama
İlaç Bilgilerini Görüntüleme	Müşteri, geçmişte aldığı ilaçların bilgilerini görüntüleyebilir.
Stok Kontrolü	Müşteri, almak istediği ilacın stok durumunu kontrol edebilir.
Rezervasyon yapma	Müşteri, istediği ilacı rezerve ederek stoktan ayırtabilir.
Fiyat Bilgisi Görüntüleme	Müşteri, ilaçların fiyatlarını görüntüleyebilir.

İlaç Gereksinimleri

Gereksinim	Açıklama
İlaç Bilgilerini Görüntüleme	İlaçların isim, fiyat, stok miktarı ve son kullanma tarihi gibi bilgileri görüntülenebilir.
Stok Güncelleme	İlaç stok miktarları artırılabilir veya azaltılabilir.
Fiyat Güncelleme	İlaçların satış ve alış fiyatları düzenlenebilir.
Tedarikçi Bazlı Görüntüleme	İlaçların hangi tedarikçiden temin edildiği görüntülenebilir.

Rezervasyon Gereksinimleri

Gereksinim	Açıklama
Rezervasyon Durum Kontrolü	Müşterilerin yaptığı rezervasyonların onay durumu görüntülenebilir.
Rezervasyon Oluşturma	Müşteri için ilaç rezervasyonu yapılabilir.
Rezervasyon Güncelleme	Müşteri veya eczacı tarafından rezervasyon bilgileri düzenlenebilir.

Stok Geçmişi Gereksinimleri

Gereksinim	Açıklama
Stok Değişimlerini Görüntüleme	İlaçların stok değişiklikleri ve tarihleri görüntülenebilir.
Stok Güncelleme Takibi	Hangi eczacının hangi stok güncellemesini yaptığı takip edilebilir.

Tedarikçi Gereksinimleri

Gereksinim	Açıklama
Tedarikçi Bilgilerini Görüntüleme	Tedarikçi iletişim ve adres bilgileri görüntülenebilir.
Tedarikçi Bazlı Stok Takibi	Tedarikçilerden alınan ilaçların stok ve teslim tarihleri takip edilebilir.
Yeni Tedarikçi Ekleme	Sisteme yeni bir tedarikçi kaydı yapılabilir.

Şube Gereksinimleri

Gereksinim	Açıklama
Şube Bilgilerini Görüntüleme	Şubenin adres, telefon ve çalışma saatleri bilgileri görüntülenebilir.
Şube İlaç Stoklarını Görüntüleme	Şubede bulunan ilaçların stok durumları kontrol edilebilir.
Şube Performans Raporu	Şubede yapılan satışlar ve işlemlerle ilgili performans raporu alınabilir.

VARLIK VE İLİŞKİLER

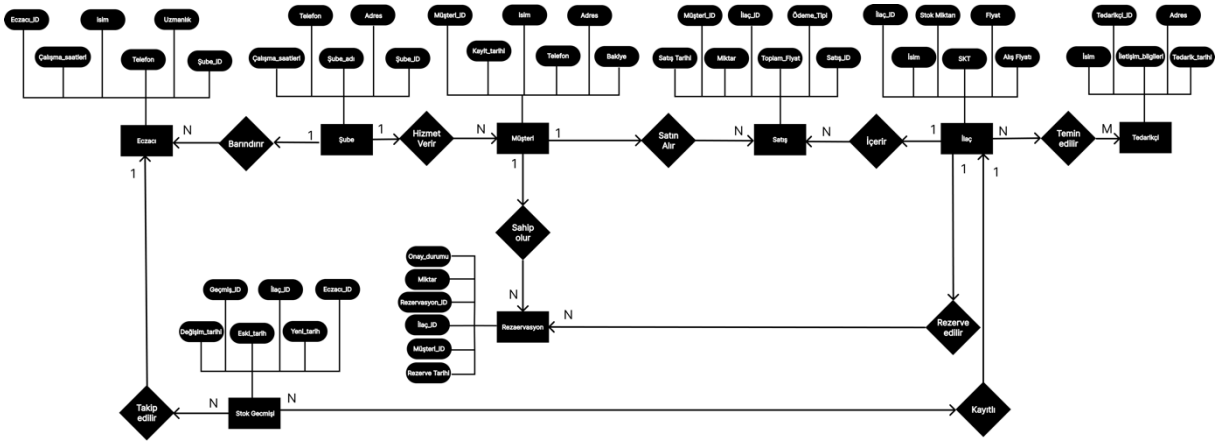
VARLIKLAR

Varlık	Özellikler	İlişkiler
Müşteri	Müşteri_ID, İsim, Telefon, Adres, Kayıt Tarihi, Bakiye	- Sahip Olur (Rezervasyon): 1:N - Satın Alır (Satış): 1:N
İlaç	İlaç_ID, İsim, Stok Miktarı, Son Kullanma Tarihi, Fiyat, Alış Fiyatı	- Satılır (Satış): 1:N - Rezerve Edilir (Rezervasyon): 1:N - Temin Edilir (Tedarikçi): N:M - Bulunur (Şube): N:M
Tedarikçi	Tedarikçi_ID, İsim, İletişim Bilgileri, Adres, Tedarik Tarihi	- Sağlar (İlaç): N:M
Satış	Satış_ID, Müşteri_ID, İlaç_ID, Satış Tarihi, Miktar, Toplam Fiyat, Ödeme Tipi	- Yapılır (Müşteri): N:1 - İçerir (İlaç): N:1
Rezervasyon	Rezervasyon_ID, Müşteri_ID, İlaç_ID, Rezervasyon Tarihi, Miktar, Onay Durumu	- Yapar (Müşteri): N:1 - İçerir (İlaç): N:1
Eczacı	Eczacı_ID, İsim, Çalışma Saatleri, Telefon, Uzmanlık, Şube_ID	- Yönetir (Stok): 1:N - Hazırlar (Rapor): 1:N
Stok Geçmişi	Geçmiş_ID, İlaç_ID, Değişim Tarihi, Eski_Miktar, Yeni Miktar, Eczacı_ID	-Kayıtlı (İlaç): N:1 -Takip Edilir (Eczacı): N:1
Şube	Şube_ID, Şube Adı, Adres, Telefon, Çalışma Saatleri	-Barındırır (Eczacı): 1:N -Hizmet Verir (Müşteri): 1:N -Barındırır (İlaç): N:M

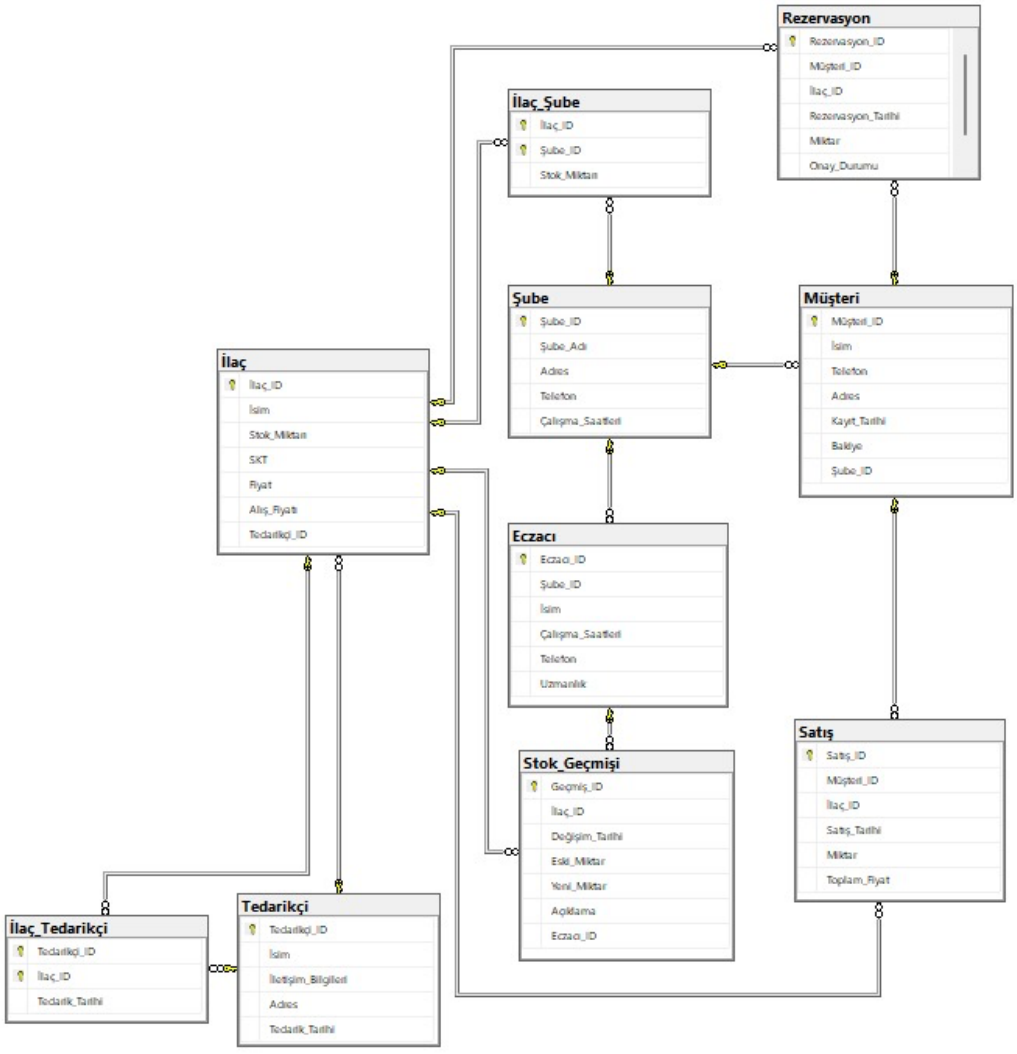
N-M İlişki Tabloları

İlişki Tablosu	Özellikler	Açıklama
İlaç_Tedarikçi	İlaç_ID, Tedarikçi_ID	Bir ilaç birden fazla tedarikçi tarafından sağlanabilir ve bir tedarikçi birçok ilacı temin edebilir.
İlaç_Şube	İlaç_ID, Şube_ID	Bir ilaç birden fazla şubede bulunabilir ve bir şube birçok ilacı barındırabilir.

E-R DİYAGRAMI



DATABASE DİAGRAMS



İLİŞKİSEL VERİ MODELİ

- **Müşteri**(Müşteri_ID, İsim, Telefon, Adres, Kayıt_Tarihi, Bakiye, Şube_ID)
- **İlaç**(İlaç_ID, İsim, Stok_Miktarı, SKT, Fiyat, Alış_Fiyatı, Tedarikçi_ID)
- **Tedarikçi**(Tedarikçi_ID, İsim, İletişim_Bilgileri, Adres, Tedarik_Tarihi)
- **İlaç_Tedarikçi**(Tedarikçi_ID, İlaç_ID, Tedarik_Tarihi)
- **Satış**(Satış_ID, Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
- **Rezervasyon**(Rezervasyon_ID, Müşteri_ID, İlaç_ID, Rezervasyon_Tarihi, Miktar, Onay_Durumu)
- **Şube**(Şube_ID, Şube_Adı, Adres, Telefon, Çalışma_Saatleri)
- **Eczacı**(Eczacı_ID, Şube_ID, İsim, Çalışma_Saatleri, Telefon, Uzmanlık)
- **Stok_Geçmişi**(Geçmiş_ID, İlaç_ID, Değişim_Tarihi, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
- **İlaç_Şube**(İlaç_ID, Şube_ID, Stok_Miktarı)

SQL KODLARI AÇIKLAMALARI

(KODUN TAMAMI RAPOR SONUNDA METİN HALİNDE BULUNMAKTADIR!)

```
CREATE TABLE Müşteri (
    Müşteri_ID INT PRIMARY KEY,
    İsim NVARCHAR(50) NOT NULL,
    Telefon NVARCHAR(15) UNIQUE,
    Adres NVARCHAR(100),
    Kayıt_Tarihi DATE DEFAULT GETDATE(),
    Bakiye DECIMAL(10,2) CHECK (Bakiye >= 0),
    Şube_ID INT FOREIGN KEY REFERENCES Şube(Şube_ID)
);
```

Bu kod, müşterilerle ilgili temel bilgilerin (isim, telefon, adres, kayıt tarihi ve bakiye) düzenli bir şekilde saklanmasını sağlar. Her müşteri, benzersiz bir **Müşteri_ID** ile tanımlanır ve müşterilerin telefon numaraları da benzersiz olacak şekilde kısıtlanmıştır. **Kayıt_Tarihi** sütunu, müşterinin sisteme eklenme tarihini otomatik olarak kaydeder. **Şube_ID** sütunu ile her müşteri, bir şubeye bağlanır ve bu ilişki, **Şube** tablosundaki **Şube_ID** ile doğrulanır. Kod ayrıca, bakiyenin negatif olmamasını sağlayarak veri tutarlılığına katkıda bulunur.

```
CREATE TABLE İlaç (
    İlaç_ID INT PRIMARY KEY,
    İsim NVARCHAR(50) NOT NULL,
    Stok_Miktarı INT CHECK (Stok_Miktarı >= 0),
    SKT DATE NOT NULL,
    Fiyat DECIMAL(10,2) CHECK (Fiyat > 0),
    Alış_Fiyatı DECIMAL(10,2) CHECK (Alış_Fiyatı > 0),
    Tedarikçi_ID INT FOREIGN KEY REFERENCES Tedarikçi(Tedarikçi_ID)
);
```

Bu kod, **İlaç** tablosunu oluşturmak için tasarlanmıştır ve eczanede bulunan ilaçlarla ilgili bilgilerin saklanmasını sağlar. Her ilaç, benzersiz bir **İlaç_ID** ile tanımlanır ve adı (**İsim**) boş bırakılamaz. **Stok_Miktarı**, ilaç stoğunun sıfır veya daha fazla olmasını garanti eden bir kısıtlamaya sahiptir. İlaçların son kullanma tarihi (**SKT**) zorunlu bir alandır ve fiyat bilgileri (**Fiyat** ve **Alış_Fiyatı**) sıfırdan büyük olmalıdır. **Tedarikçi_ID** sütunu, ilacın hangi tedarikçiden temin edildiğini belirtir ve **Tedarikçi** tablosundaki **Tedarikçi_ID** sütununa bir dış anahtar ile bağlanır. Bu yapı, ilaçların tedarikçi ilişkilerini etkin bir şekilde yönetmek için kullanılır.

```sql
CREATE TABLE Tedarikçi (
    Tedarikçi_ID INT PRIMARY KEY,
    İsim NVARCHAR(50) NOT NULL,
    İletişim_Bilgileri NVARCHAR(100),
    Adres NVARCHAR(100),
    Tedarik_Tarihi DATE DEFAULT GETDATE()
);
```

Bu kod, **Tedarikçi** tablosunu oluşturmak için tasarlanmıştır ve eczaneye ilaç sağlayan tedarikçilerle ilgili bilgileri saklar. Her tedarikçi, benzersiz bir **Tedarikçi_ID** ile tanımlanır ve adı (**İsim**) boş bırakılamaz. **İletişim_Bilgileri** ve **Adres** sütunları, tedarikçiye ulaşılabilmesi için gerekli bilgileri içerir. **Tedarik_Tarihi** sütunu, tedarikçinin sisteme eklendiği tarihi varsayılan olarak kaydeder. Bu tablo, tedarikçilerin kimlik bilgilerini ve iletişim detaylarını düzenli bir şekilde tutar.

```sql
CREATE TABLE İlaç_Tedarikçi (
    Tedarikçi_ID INT FOREIGN KEY REFERENCES Tedarikçi(Tedarikçi_ID),
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Tedarik_Tarihi DATE DEFAULT GETDATE(),
    PRIMARY KEY (Tedarikçi_ID, İlaç_ID)
);
```

Bu kod, **İlaç_Tedarikçi** tablosunu oluşturur ve tedarikçiler ile ilaçlar arasındaki ilişkiyi yönetir. Tablo, her bir ilacın hangi tedarikçi tarafından sağlandığını ve bu tedarikin ne zaman yapıldığını kaydeder. **Tedarikçi_ID** sütunu, **Tedarikçi** tablosundaki **Tedarikçi_ID** ile ilişkilidir, aynı şekilde **İlaç_ID** sütunu da **İlaç** tablosundaki **İlaç_ID** ile ilişkilidir. Bu iki sütun birlikte birincil anahtarı (**PRIMARY KEY**) oluşturur, bu da aynı tedarikçi ve ilaç ilişkisinin birden fazla kez kaydedilmesini önler. **Tedarik_Tarihi** sütunu, her bir tedarik işleminin tarihini otomatik olarak kaydeder. Bu tablo, tedarik işlemlerinin detaylı bir şekilde izlenmesini sağlar.

```sql
CREATE TABLE Satış (
    Satış_ID INT IDENTITY(1,1) PRIMARY KEY,
    Müşteri_ID INT FOREIGN KEY REFERENCES Müşteri(Müşteri_ID),
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Satış_Tarihi DATE DEFAULT GETDATE(),
    Miktar INT CHECK (Miktar > 0),
    Toplam_Fiyat DECIMAL(10,2) CHECK (Toplam_Fiyat > 0)
);
```

Bu kod, **Satış** tablosunu oluşturur ve müşterilerin satın alma işlemlerini kaydetmek için tasarlanmıştır. Her satış işlemi, otomatik artan bir **Satış_ID** ile benzersiz bir şekilde tanımlanır. **Müşteri_ID** sütunu, **Müşteri** tablosundaki **Müşteri_ID** ile ve **İlaç_ID** sütunu, **İlaç** tablosundaki **İlaç_ID** ile ilişkilidir. **Satış_Tarihi** sütunu, satış işleminin gerçekleştiği tarihi otomatik olarak kaydeder. **Miktar** sütunu, satın alınan ürün miktarını belirtir ve sıfırdan büyük olmalıdır. **Toplam_Fiyat** sütunu ise toplam tutarı saklar ve sıfırdan büyük olma kuralı ile sınırlandırılmıştır. Bu yapı, her satış işlemini detaylı ve doğrulanabilir bir şekilde yönetmek için kullanılır.

```sql
CREATE TABLE Rezervasyon (
    Rezervasyon_ID INT PRIMARY KEY,
    Müşteri_ID INT FOREIGN KEY REFERENCES Müşteri(Müşteri_ID),
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Rezervasyon_Tarihi DATE DEFAULT GETDATE(),
    Miktar INT CHECK (Miktar > 0),
    Onay_Durumu NVARCHAR(20) DEFAULT 'Beklemede'
);
```

Bu kod, **Rezervasyon** tablosunu oluşturur ve müşterilerin ilaç rezervasyonlarını yönetmek için tasarlanmıştır. Her rezervasyon, benzersiz bir **Rezervasyon_ID** ile tanımlanır. **Müşteri_ID** sütunu, **Müşteri** tablosundaki müşterilere; **İlaç_ID** sütunu ise **İlaç** tablosundaki ilaçlara referans verir ve bu sütunlar **FOREIGN KEY** olarak tanımlanmıştır. **Rezervasyon_Tarihi** sütunu, rezervasyonun yapıldığı tarihi otomatik olarak kaydeder. **Miktar** sütunu, rezervasyon yapılan ilaç miktarını tutar ve sıfırdan büyük olma şartına sahiptir. **Onay_Durumu** sütunu, rezervasyonun durumunu belirtir ve varsayılan olarak 'Beklemede' değerini alır. Bu yapı, ilaç rezervasyonlarını düzenli bir şekilde takip etmek için tasarlanmıştır.

```sql
CREATE TABLE Şube (
    Şube_ID INT PRIMARY KEY,
    Şube_Adı NVARCHAR(50) NOT NULL,
    Adres NVARCHAR(100) NOT NULL,
    Telefon NVARCHAR(15),
    Çalışma_Saatleri NVARCHAR(50)
);
```

Bu kod, **Şube** tablosunu oluşturur ve eczanenin farklı şubeleriyle ilgili bilgileri tutmak için tasarlanmıştır. Her şube, benzersiz bir **Şube_ID** ile tanımlanır. **Şube_Adı** sütunu şubenin adını, **Adres** sütunu ise şubenin adres bilgilerini saklar. **Telefon** sütunu, şubenin telefon numarasını içerir ve **Çalışma_Saatleri** sütunu şubenin çalışma saatlerini belirtir. **Şube_Adı** ve **Adres** sütunları, şubenin temel bilgilerini eksiksiz sağlamak için NOT NULL olarak işaretlenmiştir. Bu tablo, eczane sistemindeki şubelerle ilgili bilgilerin merkezi bir şekilde yönetilmesine olanak tanır.

```sql
CREATE TABLE Eczacı (
    Eczacı_ID INT PRIMARY KEY,
    Şube_ID INT FOREIGN KEY REFERENCES Şube(Şube_ID),
    İsim NVARCHAR(50) NOT NULL,
    Çalışma_Saatleri NVARCHAR(50),
    Telefon NVARCHAR(15),
    Uzmanlık NVARCHAR(50)
);
```

Bu tablo, eczanelerde çalışan eczacıların bilgilerini saklamak için tasarlanmıştır. **Eczacı_ID**, her eczacıyı benzersiz bir şekilde tanımlayan birincil anahtar sütunudur. **Şube_ID**, eczacının çalıştığı şubeyi belirtir ve **Şube** tablosuyla bir dış anahtar ilişkisi kurar. **İsim** sütunu, eczacının adını içerir ve boş bırakılamaz. **Çalışma_Saatleri**, eczacının çalışma saatlerini belirtirken; **Telefon** ve **Uzmanlık** sütunları, iletişim bilgilerini ve uzmanlık alanını içerir. Bu tablo, eczacıların kayıtlarını düzenlemek ve ilgili şubeyle ilişkilerini yönetmek için kullanılır.

```sql
CREATE TABLE Stok_Geçmişi (
    Geçmiş_ID INT IDENTITY(1,1) PRIMARY KEY,
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Değişim_Tarihi DATE DEFAULT GETDATE(),
    Eski_Miktar INT,
    Yeni_Miktar INT,
    Açıklama NVARCHAR(100),
    Eczacı_ID INT FOREIGN KEY REFERENCES Eczacı(Eczacı_ID)
);
```

Bu tablo, ilaç stoklarındaki değişikliklerin kaydını tutmak için tasarlanmıştır. **Geçmiş_ID**, her stok değişikliğini benzersiz şekilde tanımlayan birincil anahtardır ve otomatik olarak artan bir değerdir. **İlaç_ID**, değişikliğin hangi ilaçla ilgili olduğunu belirtir ve **İlaç** tablosuyla bir dış anahtar ilişkisi kurar. **Değişim_Tarihi**, değişikliğin yapıldığı tarihi otomatik olarak kaydeder. **Eski_Miktar** ve **Yeni_Miktar** sütunları, stokta yapılan değişiklik öncesi ve sonrası miktarları gösterir. **Açıklama**, değişiklikle ilgili açıklama veya not eklemek için kullanılır. **Eczacı_ID**, değişikliği gerçekleştiren eczacıyı belirtir ve **Eczacı** tablosuna bağlıdır. Bu tablo, stok hareketlerini takip etmek ve geçmişe yönelik incelemeler yapmak için kritik öneme sahiptir.

```sql
CREATE TABLE İlaç_Şube (
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Şube_ID INT FOREIGN KEY REFERENCES Şube(Şube_ID),
    Stok_Miktarı INT CHECK (Stok_Miktarı >= 0),
    PRIMARY KEY (İlaç_ID, Şube_ID)
);
```

Bu tablo, eczane şubelerinde bulunan ilaçların stok durumunu yönetmek için tasarlanmıştır. **İlaç_ID**, tabloya hangi ilacın ait olduğunu belirtir ve **İlaç** tablosuyla bir dış anahtar ilişkisi kurar. **Şube_ID**, ilgili şubeyi temsil eder ve **Şube** tablosuna bağlanır. **Stok_Miktarı**, o şubede bulunan ilaç miktarını belirtir ve 0 veya daha büyük bir değer olması gerekir (CHECK kısıtlaması ile doğrulanır). **PRIMARY KEY (İlaç_ID, Şube_ID)** ile her ilaç ve şube kombinasyonu benzersiz şekilde tanımlanır. Bu tablo, ilaçların şubeler bazında dağılımını ve mevcut stok durumunu takip etmek için kritik öneme sahiptir.

```sql
CREATE TRIGGER SatışSonrasıStokGuncelle
ON Satış
AFTER INSERT
AS
BEGIN
    BEGIN TRY
        -- Yeni eklenen satış kaydındaki İlaç_ID ve Miktar değerlerini çek
        DECLARE @İlaç_ID INT, @Miktar INT;

        SELECT
            @İlaç_ID = İlaç_ID,
            @Miktar = Miktar
        FROM inserted;

        -- Stok güncelleme prosedürünü çağır
        EXEC StokGuncelle @İlaç_ID, @Miktar;

    END TRY
    BEGIN CATCH
        -- Hata oluşursa işlem iptal edilmez, sadece mesaj yazdırılır
        PRINT 'Stok güncellemesi sırasında bir hata oluştu: ' + ERROR_MESSAGE();
    END CATCH
END;
```

Bu tetikleyici, **Satış** tablosuna yeni bir satış kaydı eklendiğinde otomatik olarak çalışır ve ilgili ilacın stok miktarını günceller. Yeni eklenen satış kaydından **İlaç_ID** ve **Miktar** bilgilerini alır ve bu bilgileri kullanarak **StokGuncelle** adlı saklı yordamı çağırır. Bu saklı yordam, stok miktarını günceller ve stok değişikliklerini **Stok_Geçmişi** tablosuna kaydeder. Eğer işlem sırasında bir hata oluşursa, tetikleyici hata mesajını yazdırır ancak işlemi iptal etmez. Bu yapı, hata toleransını artırarak satış işlemlerinin kesintiye uğramadan devam etmesini sağlar.

```sql
CREATE OR ALTER PROCEDURE StokGuncelle

    @İlaç_ID INT,
    @Miktar INT
AS
BEGIN
    -- Stok Güncellemesi
    UPDATE İlaç
    SET Stok_Miktarı = Stok_Miktarı - @Miktar
    WHERE İlaç_ID = @İlaç_ID;

    -- Stok Geçmişine Kayıt (Geçmiş_ID otomatik artacak)
    INSERT INTO Stok_Geçmişi (İlaç_ID, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
    VALUES (
        @İlaç_ID,
        (SELECT Stok_Miktarı + @Miktar FROM İlaç WHERE İlaç_ID = @İlaç_ID),
        (SELECT Stok_Miktarı FROM İlaç WHERE İlaç_ID = @İlaç_ID),
        'Satış',
        NULL
    );
END;
```

Bu saklı yordam (Stored Procedure), verilen **İlaç_ID** ve **Miktar** değerlerini kullanarak ilgili ilacın stok miktarını günceller. İlk olarak, **İlaç** tablosundaki **Stok_Miktarı**, belirtilen miktar kadar azaltılır. Ardından, bu güncelleme bilgileri **Stok_Geçmişi** tablosuna kaydedilir. Kayıtta, ilacın eski stok miktarı, yeni stok miktarı, işlem açıklaması (örneğin, "Satış") ve gerekirse hangi eczacının bu işlemi gerçekleştirdiğine dair bilgiler eklenir. Bu yordam, stok yönetimini otomatikleştirir ve işlem geçmişini izlenebilir hale getirir. Böylece hem stokların doğru yönetilmesi hem de sistemdeki şeffaflık sağlanmış olur.

```sql
CREATE OR ALTER PROCEDURE StokEkle
    @İlaç_ID INT,
    @EklenecekMiktar INT
AS
BEGIN
    -- Mevcut stok miktarını artır
    UPDATE İlaç
    SET Stok_Miktarı = Stok_Miktarı + @EklenecekMiktar
    WHERE İlaç_ID = @İlaç_ID;

    -- Stok Geçmişine bu işlemi kaydet
    INSERT INTO Stok_Geçmişi (İlaç_ID, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
    VALUES (
        @İlaç_ID,
        (SELECT Stok_Miktarı - @EklenecekMiktar FROM İlaç WHERE İlaç_ID = @İlaç_ID),
        (SELECT Stok_Miktarı FROM İlaç WHERE İlaç_ID = @İlaç_ID),
        'Stok Eklendi',
        NULL
    );
END;
```

Bu saklı yordam (Stored Procedure), verilen **İlaç_ID** ve **EklenecekMiktar** parametrelerini kullanarak ilgili ilacın stok miktarını artırır. İlk olarak, **İlaç** tablosundaki **Stok_Miktarı**, belirtilen miktar kadar güncellenir (artırılır). Ardından, bu güncelleme işlemi **Stok_Geçmişi** tablosuna kaydedilir. Kayıt sırasında ilacın eski stok miktarı, yeni stok miktarı, işlem açıklaması ("Stok Eklendi") ve işlem yapan eczacı bilgisi (eğer varsa) eklenir. Bu yordam, stok ekleme işlemlerini düzenli ve kayıtlı bir şekilde gerçekleştirerek stok yönetiminde şeffaflık ve izlenebilirlik sağlar.

```sql
BEGIN TRANSACTION; -- Satış işlemi sırasında hata tespit edilecek
BEGIN TRY
    -- Satış kaydı eklenir
    INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
    VALUES (1, 1, GETDATE(), 2, 40.00);

    -- Stok güncelleme prosedürü çağrılır
    EXEC StokGuncelle @İlaç_ID = 1, @Miktar = 2;

    -- İşlem başarılı, commit edilir
    COMMIT;
END TRY
BEGIN CATCH
    -- Hata durumunda işlem geri alınır
    ROLLBACK;
    PRINT 'Transaction başarısız: ' + ERROR_MESSAGE();
END CATCH;
```

Bu işlem, SQL Server'da bir satış işleminin güvenli bir şekilde tamamlanmasını sağlamak için bir **transaction** (işlem) yapısını kullanır. İlk olarak, bir **BEGIN TRANSACTION** ifadesiyle işlem başlatılır. **TRY** bloğu içinde, satış kaydı **Satış** tablosuna eklenir ve ardından stok miktarını azaltan **StokGuncelle** prosedürü çağrılır. Eğer her iki işlem de başarılı olursa, **COMMIT** ifadesiyle işlem kalıcı hale getirilir. Ancak bir hata meydana gelirse, **CATCH** bloğu devreye girer ve **ROLLBACK** ile yapılan tüm değişiklikler geri alınır. Bu yapı, tutarlılığı korumak ve hatalı işlemlerin veri tabanında değişiklik yapmasını engellemek için tasarlanmıştır.

```sql
BEGIN TRANSACTION; -- Stok yetersizliği kontrol edilir
BEGIN TRY
    -- Satış kaydı eklenir
    INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
    VALUES (1, 1, GETDATE(), 5000, 200000.00); -- Hata: Stok yetersiz

    -- Stok güncelleme prosedürü çağrılır
    EXEC StokGuncelle @İlaç_ID = 1, @Miktar = 5000;

    -- İşlem başarılı, commit edilir
    COMMIT;
END TRY
BEGIN CATCH
    -- Hata durumunda işlem geri alınır
    ROLLBACK;
    PRINT 'Stok yetersiz, işlem iptal edildi: ' + ERROR_MESSAGE();
END CATCH;
```

Bu işlem, stok yetersizliği gibi durumlarda veri tabanı tutarlılığını sağlamak için bir **transaction** yapısını kullanır. İşlem **BEGIN TRANSACTION** ile başlar ve **TRY** bloğu içinde önce satış kaydı eklenir. Daha sonra, stok miktarını azaltan **StokGuncelle** prosedürü çağrılır. Eğer işlem sırasında stok yetersizliği gibi bir hata meydana gelirse, **CATCH** bloğu devreye girer. **ROLLBACK** ile yapılan değişiklikler geri alınır ve kullanıcıya "Stok yetersiz, işlem iptal edildi" mesajı gösterilir. Bu yapı, hata durumunda veri tabanının tutarlılığını koruyarak herhangi bir yanlış veri girişini önler.

ÖRNEK VERİLERİN EKLENMESİ İLE İLGİLİ KISIM

```sql
-- ÖRNEK VERİLER

-- Şube Verileri
INSERT INTO Şube VALUES (1, 'Merkez Şube', 'Atatürk Caddesi No:12, Ankara', '0312-1234567', '08:00 - 18:00');
INSERT INTO Şube VALUES (2, 'Kadıköy Şube', 'Bağdat Caddesi No:45, İstanbul', '0216-9876543', '09:00 - 19:00');
INSERT INTO Şube VALUES (3, 'İzmir Şube', 'Kordon No:5, İzmir', '0232-6543210', '10:00 - 20:00');

-- Eczacı Verileri
INSERT INTO Eczacı VALUES (1, 1, 'Mehmet Demir', '08:00 - 17:00', '0532-1112233', 'Dermatoloji');
INSERT INTO Eczacı VALUES (2, 2, 'Ayşe Yılmaz', '09:00 - 18:00', '0543-4445566', 'Pediatri');
INSERT INTO Eczacı VALUES (3, 3, 'Ali Kara', '10:00 - 19:00', '0555-7778899', 'Genel Eczacılık');

-- Müşteri Verileri
INSERT INTO Müşteri VALUES (1, 'Ahmet Yılmaz', '555-1234', 'İstanbul', '2024-01-01', 1000.00, 1);
INSERT INTO Müşteri VALUES (2, 'Fatma Demir', '555-5678', 'Ankara', '2024-02-01', 800.00, 2);
INSERT INTO Müşteri VALUES (3, 'Hasan Kaya', '555-9012', 'İzmir', '2024-03-01', 500.00, 3);

-- Tedarikçi Verileri
INSERT INTO Tedarikçi VALUES (101, 'İlaç A.Ş.', '123456789', 'Ankara', '2024-01-01');
INSERT INTO Tedarikçi VALUES (102, 'Sağlık Ltd.', '987654321', 'İstanbul', '2024-02-01');
INSERT INTO Tedarikçi VALUES (103, 'Medikal İlaç Sanayi', '456789123', 'İzmir', '2024-03-01');

-- İlaç Verileri
INSERT INTO İlaç VALUES (1, 'Aspirin', 100, '2025-12-31', 20.00, 10.00, 101);
INSERT INTO İlaç VALUES (2, 'Parol', 200, '2024-12-31', 15.00, 8.00, 102);
INSERT INTO İlaç VALUES (3, 'C vitamini', 150, '2026-06-30', 25.00, 12.00, 103);
-- İlaç_Tedarikçi Verileri
INSERT INTO İlaç_Tedarikçi VALUES (101, 1, '2024-01-01');
INSERT INTO İlaç_Tedarikçi VALUES (102, 2, '2024-02-01');
INSERT INTO İlaç_Tedarikçi VALUES (103, 3, '2024-03-01');

-- Rezervasyon Verileri
INSERT INTO Rezervasyon VALUES (1, 1, 1, '2024-01-06', 2, 'Beklemede');
INSERT INTO Rezervasyon VALUES (2, 2, 2, '2024-02-06', 1, 'Onaylandı');
INSERT INTO Rezervasyon VALUES (3, 3, 3, '2024-03-06', 3, 'Beklemede');

-- Satış Verileri
INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (1, 1, '2024-01-05', 2, 40.00);
INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (2, 2, '2024-02-05', 1, 15.00);
INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (3, 3, '2024-03-05', 3, 75.00);

-- Stok Geçmişi Verileri
INSERT INTO Stok_Geçmişi (İlaç_ID, Değişim_Tarihi, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
VALUES
(1, '2024-01-05', 100, 98, 'Satış', 1),
(2, '2024-02-05', 200, 199, 'Satış', 2),
(3, '2024-03-05', 150, 147, 'Satış', 3);

-- İlaç_Şube Verileri
INSERT INTO İlaç_Şube VALUES (1, 1, 50);
INSERT INTO İlaç_Şube VALUES (2, 2, 75);
INSERT INTO İlaç_Şube VALUES (3, 3, 30);
```

GEREKLİ TESTLERİN YAPILMASI

```sql
SELECT * FROM Müşteri;  -- Tabloların hepsini inceledik
SELECT * FROM İlaç;
SELECT * FROM İlaç_Tedarikçi;
SELECT * FROM İlaç_Şube;
SELECT * FROM Tedarikçi;
SELECT * FROM Satış;
SELECT * FROM Rezervasyon;
SELECT * FROM Stok_Geçmişi;
SELECT * FROM Eczacı;
SELECT * FROM Şube;

EXEC StokGuncelle @İlaç_ID = 1, @Miktar = 10;  -- Stok azaltma test edildi

EXEC StokEkle @İlaç_ID = 1, @EklenecekMiktar = 10;  -- Stok ekleme test edildi

INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (1, 1, GETDATE(), 5, 100.00); -- Trigger kontrolü
```

SQL KODUNUN TAMAMI

```
CREATE TABLE Müşteri (
   Müşteri_ID INT PRIMARY KEY,
   İsim NVARCHAR(50) NOT NULL,
   Telefon NVARCHAR(15) UNIQUE,
   Adres NVARCHAR(100),
   Kayıt_Tarihi DATE DEFAULT GETDATE(),
   Bakiye DECIMAL(10,2) CHECK (Bakiye >= 0),
   Şube_ID INT FOREIGN KEY REFERENCES Şube(Şube_ID)
);

CREATE TABLE İlaç (
   İlaç_ID INT PRIMARY KEY,
   İsim NVARCHAR(50) NOT NULL,
   Stok_Miktarı INT CHECK (Stok_Miktarı >= 0),
   SKT DATE NOT NULL,
   Fiyat DECIMAL(10,2) CHECK (Fiyat > 0),
   Alış_Fiyatı DECIMAL(10,2) CHECK (Alış_Fiyatı > 0),
   Tedarikçi_ID INT FOREIGN KEY REFERENCES Tedarikçi(Tedarikçi_ID)
);

CREATE TABLE Tedarikçi (
   Tedarikçi_ID INT PRIMARY KEY,
   İsim NVARCHAR(50) NOT NULL,
   İletişim_Bilgileri NVARCHAR(100),
   Adres NVARCHAR(100),
   Tedarik_Tarihi DATE DEFAULT GETDATE()
);
```

```
CREATE TABLE İlaç_Tedarikçi (
  Tedarikçi_ID INT FOREIGN KEY REFERENCES Tedarikçi(Tedarikçi_ID),
  İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
  Tedarik_Tarihi DATE DEFAULT GETDATE(),
  PRIMARY KEY (Tedarikçi_ID, İlaç_ID)
);

CREATE TABLE Satış (
  Satış_ID INT IDENTITY(1,1) PRIMARY KEY,
  Müşteri_ID INT FOREIGN KEY REFERENCES Müşteri(Müşteri_ID),
  İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
  Satış_Tarihi DATE DEFAULT GETDATE(),
  Miktar INT CHECK (Miktar > 0),
  Toplam_Fiyat DECIMAL(10,2) CHECK (Toplam_Fiyat > 0)
);

CREATE TABLE Rezervasyon (
  Rezervasyon_ID INT PRIMARY KEY,
  Müşteri_ID INT FOREIGN KEY REFERENCES Müşteri(Müşteri_ID),
  İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
  Rezervasyon_Tarihi DATE DEFAULT GETDATE(),
  Miktar INT CHECK (Miktar > 0),
  Onay_Durumu NVARCHAR(20) DEFAULT 'Beklemede'
);

CREATE TABLE Şube (
  Şube_ID INT PRIMARY KEY,
  Şube_Adı NVARCHAR(50) NOT NULL,
```

```
    Adres NVARCHAR(100) NOT NULL,
    Telefon NVARCHAR(15),
    Çalışma_Saatleri NVARCHAR(50)
);

CREATE TABLE Eczacı (
    Eczacı_ID INT PRIMARY KEY,
    Şube_ID INT FOREIGN KEY REFERENCES Şube(Şube_ID),
    İsim NVARCHAR(50) NOT NULL,
    Çalışma_Saatleri NVARCHAR(50),
    Telefon NVARCHAR(15),
    Uzmanlık NVARCHAR(50)
);

CREATE TABLE Stok_Geçmişi (
    Geçmiş_ID INT IDENTITY(1,1) PRIMARY KEY,
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Değişim_Tarihi DATE DEFAULT GETDATE(),
    Eski_Miktar INT,
    Yeni_Miktar INT,
    Açıklama NVARCHAR(100),
    Eczacı_ID INT FOREIGN KEY REFERENCES Eczacı(Eczacı_ID)
);

CREATE TABLE İlaç_Şube (
    İlaç_ID INT FOREIGN KEY REFERENCES İlaç(İlaç_ID),
    Şube_ID INT FOREIGN KEY REFERENCES Şube(Şube_ID),
    Stok_Miktarı INT CHECK (Stok_Miktarı >= 0),
    PRIMARY KEY (İlaç_ID, Şube_ID)
```

```
);

-- ÖRNEK VERİLER

-- Şube Verileri
INSERT INTO Şube VALUES (1, 'Merkez Şube', 'Atatürk Caddesi No:12, Ankara', '0312-1234567', '08:00 - 18:00');
INSERT INTO Şube VALUES (2, 'Kadıköy Şube', 'Bağdat Caddesi No:45, İstanbul', '0216-9876543', '09:00 - 19:00');
INSERT INTO Şube VALUES (3, 'İzmir Şube', 'Kordon No:5, İzmir', '0232-6543210', '10:00 - 20:00');

-- Eczacı Verileri
INSERT INTO Eczacı VALUES (1, 1, 'Mehmet Demir', '08:00 - 17:00', '0532-1112233', 'Dermatoloji');
INSERT INTO Eczacı VALUES (2, 2, 'Ayşe Yılmaz', '09:00 - 18:00', '0543-4445566', 'Pediatri');
INSERT INTO Eczacı VALUES (3, 3, 'Ali Kara', '10:00 - 19:00', '0555-7778899', 'Genel Eczacılık');

-- Müşteri Verileri
INSERT INTO Müşteri VALUES (1, 'Ahmet Yılmaz', '555-1234', 'İstanbul', '2024-01-01', 1000.00, 1);
INSERT INTO Müşteri VALUES (2, 'Fatma Demir', '555-5678', 'Ankara', '2024-02-01', 800.00, 2);
INSERT INTO Müşteri VALUES (3, 'Hasan Kaya', '555-9012', 'İzmir', '2024-03-01', 500.00, 3);

-- Tedarikçi Verileri
INSERT INTO Tedarikçi VALUES (101, 'İlaç A.Ş.', '123456789', 'Ankara', '2024-01-01');
INSERT INTO Tedarikçi VALUES (102, 'Sağlık Ltd.', '987654321', 'İstanbul', '2024-02-01');
INSERT INTO Tedarikçi VALUES (103, 'Medikal İlaç Sanayi', '456789123', 'İzmir', '2024-03-01');

-- İlaç Verileri
INSERT INTO İlaç VALUES (1, 'Aspirin', 100, '2025-12-31', 20.00, 10.00, 101);
INSERT INTO İlaç VALUES (2, 'Parol', 200, '2024-12-31', 15.00, 8.00, 102);
INSERT INTO İlaç VALUES (3, 'C vitamini', 150, '2026-06-30', 25.00, 12.00, 103);
```

```
-- İlaç_Tedarikçi Verileri
INSERT INTO İlaç_Tedarikçi VALUES (101, 1, '2024-01-01');
INSERT INTO İlaç_Tedarikçi VALUES (102, 2, '2024-02-01');
INSERT INTO İlaç_Tedarikçi VALUES (103, 3, '2024-03-01');

-- Rezervasyon Verileri
INSERT INTO Rezervasyon VALUES (1, 1, 1, '2024-01-06', 2, 'Beklemede');
INSERT INTO Rezervasyon VALUES (2, 2, 2, '2024-02-06', 1, 'Onaylandı');
INSERT INTO Rezervasyon VALUES (3, 3, 3, '2024-03-06', 3, 'Beklemede');

-- Satış Verileri
INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (1, 1, '2024-01-05', 2, 40.00);
INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (2, 2, '2024-02-05', 1, 15.00);
INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
VALUES (3, 3, '2024-03-05', 3, 75.00);

-- Stok Geçmişi Verileri
INSERT INTO Stok_Geçmişi (İlaç_ID, Değişim_Tarihi, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
VALUES
(1, '2024-01-05', 100, 98, 'Satış', 1),
(2, '2024-02-05', 200, 199, 'Satış', 2),
(3, '2024-03-05', 150, 147, 'Satış', 3);

-- İlaç_Şube Verileri
INSERT INTO İlaç_Şube VALUES (1, 1, 50);
INSERT INTO İlaç_Şube VALUES (2, 2, 75);
```

```
INSERT INTO İlaç_Şube VALUES (3, 3, 30);


CREATE OR ALTER PROCEDURE StokGuncelle
  @İlaç_ID INT,
  @Miktar INT
AS
BEGIN
  -- Stok Güncellemesi
  UPDATE İlaç
  SET Stok_Miktarı = Stok_Miktarı - @Miktar
  WHERE İlaç_ID = @İlaç_ID;


  -- Stok Geçmişine Kayıt (Geçmiş_ID otomatik artacak)
  INSERT INTO Stok_Geçmişi (İlaç_ID, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
  VALUES (
    @İlaç_ID,
    (SELECT Stok_Miktarı + @Miktar FROM İlaç WHERE İlaç_ID = @İlaç_ID),
    (SELECT Stok_Miktarı FROM İlaç WHERE İlaç_ID = @İlaç_ID),
    'Satış',
    NULL
  );
END;


CREATE OR ALTER PROCEDURE StokEkle
  @İlaç_ID INT,
  @EklenecekMiktar INT
AS
BEGIN
  -- Mevcut stok miktarını artır
```

```
  UPDATE İlaç
  SET Stok_Miktarı = Stok_Miktarı + @EklenecekMiktar
  WHERE İlaç_ID = @İlaç_ID;

  -- Stok Geçmişine bu işlemi kaydet
  INSERT INTO Stok_Geçmişi (İlaç_ID, Eski_Miktar, Yeni_Miktar, Açıklama, Eczacı_ID)
  VALUES (
    @İlaç_ID,
    (SELECT Stok_Miktarı - @EklenecekMiktar FROM İlaç WHERE İlaç_ID = @İlaç_ID),
    (SELECT Stok_Miktarı FROM İlaç WHERE İlaç_ID = @İlaç_ID),
    'Stok Eklendi',
    NULL
  );
END;

CREATE TRIGGER SatışSonrasıStokGuncelle
ON Satış
AFTER INSERT
AS
BEGIN
  BEGIN TRY
    -- Yeni eklenen satış kaydındaki İlaç_ID ve Miktar değerlerini çek
    DECLARE @İlaç_ID INT, @Miktar INT;

    SELECT
      @İlaç_ID = İlaç_ID,
      @Miktar = Miktar
    FROM inserted;
```

```
    -- Stok güncelleme prosedürünü çağır
    EXEC StokGuncelle @İlaç_ID, @Miktar;

  END TRY
  BEGIN CATCH
    -- Hata oluşursa işlem iptal edilmez, sadece mesaj yazdırılır
    PRINT 'Stok güncellemesi sırasında bir hata oluştu: ' + ERROR_MESSAGE();
  END CATCH
END;

BEGIN TRANSACTION; -- Satış işlemi sırasında hata tespit edilecek
BEGIN TRY
  -- Satış kaydı eklenir
  INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
  VALUES (1, 1, GETDATE(), 2, 40.00);

  -- Stok güncelleme prosedürü çağrılır
  EXEC StokGuncelle @İlaç_ID = 1, @Miktar = 2;

  -- İşlem başarılı, commit edilir
  COMMIT;
END TRY
BEGIN CATCH
  -- Hata durumunda işlem geri alınır
  ROLLBACK;
  PRINT 'Transaction başarısız: ' + ERROR_MESSAGE();
END CATCH;

BEGIN TRANSACTION; -- Stok yetersizliği kontrol edilir
```

```
BEGIN TRY
  -- Satış kaydı eklenir
  INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)
  VALUES (1, 1, GETDATE(), 5000, 200000.00); -- Hata: Stok yetersiz

  -- Stok güncelleme prosedürü çağrılır
  EXEC StokGuncelle @İlaç_ID = 1, @Miktar = 5000;

  -- İşlem başarılı, commit edilir
  COMMIT;
END TRY
BEGIN CATCH
  -- Hata durumunda işlem geri alınır
  ROLLBACK;
  PRINT 'Stok yetersiz, işlem iptal edildi: ' + ERROR_MESSAGE();
END CATCH;

SELECT * FROM Müşteri;  -- Tabloların hepsini inceledik
SELECT * FROM İlaç;
SELECT * FROM İlaç_Tedarikçi;
SELECT * FROM İlaç_Şube;
SELECT * FROM Tedarikçi;
SELECT * FROM Satış;
SELECT * FROM Rezervasyon;
SELECT * FROM Stok_Geçmişi;
SELECT * FROM Eczacı;
SELECT * FROM Şube;

EXEC StokGuncelle @İlaç_ID = 1, @Miktar = 10;  -- Stok azaltma test edildi
```

```
EXEC StokEkle @İlaç_ID = 1, @EklenecekMiktar = 10;  -- Stok ekleme test edildi


INSERT INTO Satış (Müşteri_ID, İlaç_ID, Satış_Tarihi, Miktar, Toplam_Fiyat)

VALUES (1, 1, GETDATE(), 5, 100.00); -- Trigger kontrolü
```