T.C.
FIRAT ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BMÜ 329 - VERİ TABANI SİSTEMLERİ
DÖNEM SONU PROJESİ

FATURA YÖNETİM SİSTEMİ

Hazırlayanlar
Fatma Aleyna DOĞAN – 220260052
İkra ŞAHİN - 230260192
Gökhan GENÇCAN - 220260150

2025

İçindekiler

1. PROJE TANITIMI VE GEREKSİNİMLERİ

Bu proje, kullanıcıların fatura bilgilerini kolayca yönetebileceği, servis sağlayıcıların ise hizmet verdikleri kullanıcılara ait faturaları dijital olarak oluşturup yönetebileceği bir Fatura Yönetim Sistemi geliştirmeyi hedeflemektedir. Sistem, elektrik, su, internet gibi farklı hizmet sağlayıcıların oluşturduğu faturaların kullanıcılar tarafından görüntülenebilmesi, ödenebilmesi ve ödeme geçmişlerinin takip edilebilmesi için bir platform sunmaktadır. Bunun yanı sıra, sistem kullanıcıları fatura durumu, son ödeme tarihi, ödeme hatırlatmaları gibi konularda bilgilendirmek için bildirimler göndermekte ve her fatura için bir fiş (makbuz) oluşturarak ödeme geçmişinin belgelenmesini sağlamaktadır.

Bu kapsamlı yapı sayesinde, kullanıcılar fatura işlemlerini tek bir platform üzerinden kolayca yönetebilirken, servis sağlayıcılar ise müşterilerine yönelik fatura ve bildirim hizmetlerini dijital ortamda düzenleyebilir. Sistem, hem kullanıcı deneyimini hem de hizmet sağlayıcı verimliliğini artırmayı hedeflemektedir.

2. VARLIKLAR (Entities)

Kullanıcı (USER):

Açıklama: Sistemi kullanan bireylerin verilerini tutar. Her kullanıcı kendi adına kayıtlı faturaları görüntüleyebilir, ödemelerini yapabilir ve geçmiş işlemlerini takip edebilir.
Nitelikler: id, name, email, password, phone_number, address.

Servis Sağlayıcı (SERVICE_PROVIDER):

Açıklama: Fatura oluşturan hizmet sağlayıcıların (ör. elektrik, su, internet) bilgilerini tutar.
Nitelikler: id, name, service_type, contact_info, unit_price.

Fatura (BILL):

Açıklama: Kullanıcılara ait faturaların bilgilerini içerir. Fatura tutarı, ödeme durumu ve diğer detaylar bu varlıkta yer alır.
Nitelikler: id, amount, due_date, paid_date, status, billing_unit, unit_usage, user_id, service_provider_id.

Ödeme (PAYMENT):

Açıklama: Fatura ödemelerini kaydeder. Bir fatura birden fazla ödeme ile kapatılabilir (ör. kısmi ödemeler).
Nitelikler: id, amount_paid, payment_date, payment_method, bill_id, user_id.

Bildirim (NOTIFICATION):

Açıklama: Kullanıcılara yapılan bilgilendirmeleri içerir. Ödeme hatırlatmaları veya gecikme bildirimleri gibi içerikler bu varlıkta tutulur.
Nitelikler: id, message, notification_type, user_id, is_read.

Fiş (INVOICE):

Açıklama: Her fatura için üretilen belgeyi temsil eder. Kullanıcılar ödeme sonrası bu belgelere erişebilir.
Nitelikler: id, file_path, bill_id.

3. ENUM'LAR (Enums)

Service_Type: Servis sağlayıcı türlerini belirtir (ör. Elektrik, Su, İnternet).

Bill_Status: Faturanın ödeme durumunu belirtir (ör. Ödenmiş, Ödenmemiş, Kısmi Ödenmiş).

Payment_Method: Ödeme yöntemini belirtir (ör. Kredi Kartı, Banka Transferi, Nakit).

Billing_Unit: Hizmet türüne bağlı olarak birim tipini belirtir (ör. kWh, m^3).

Notification_Type: Bildirim türlerini ifade eder (ör. SMS, E-posta, Uygulama Bildirimi).

4. VARLIK İLİŞKİLERİ

Kullanıcı ve Fatura (USER - BILL):

İlişki: 1:N
Açıklama: Bir kullanıcı birden fazla faturaya sahip olabilir, ancak her fatura yalnızca bir kullanıcıya aittir.

Servis Sağlayıcı ve Fatura (SERVICE_PROVIDER - BILL):

İlişki: 1:N
Açıklama: Bir servis sağlayıcı birden fazla fatura oluşturabilir, ancak her fatura yalnızca bir sağlayıcıya bağlıdır.

Fatura ve Ödeme (BILL - PAYMENT):

İlişki: 1:N
Açıklama: Bir fatura birden fazla ödeme ile kapatılabilir.

Kullanıcı ve Ödeme (USER - PAYMENT):

İlişki: 1:N
Açıklama: Bir kullanıcı birden fazla ödeme yapabilir.

Kullanıcı ve Bildirim (USER - NOTIFICATION):

İlişki: 1:N
Açıklama: Bir kullanıcı birden fazla bildirim alabilir.

Fatura ve Fiş (BILL - INVOICE):

İlişki: 1:1
Açıklama: Her fatura için yalnızca bir fiş oluşturulur.

5. ROL TABANLI GEREKSİNİMLER

1. Müşteri (Kullanıcı) Gereksinimleri

Fatura görüntüleme: Kullanıcılar yalnızca kendilerine ait faturaları görebilmelidir.
Fatura ödeme: Kullanıcılar faturalarını sistem üzerinden çeşitli ödeme yöntemleriyle ödeyebilmelidir.
Ödeme geçmişini görüntüleme: Kullanıcılar geçmişte yaptıkları ödemeleri ve durumlarını inceleyebilmelidir.
Bildirimleri takip etme: Fatura hatırlatmaları ve gecikme bildirimlerini görüntüleyebilmelidir.
Fişlere erişim sağlama: Ödenen faturalar için oluşturulan fişleri indirip görüntüleyebilmelidir.

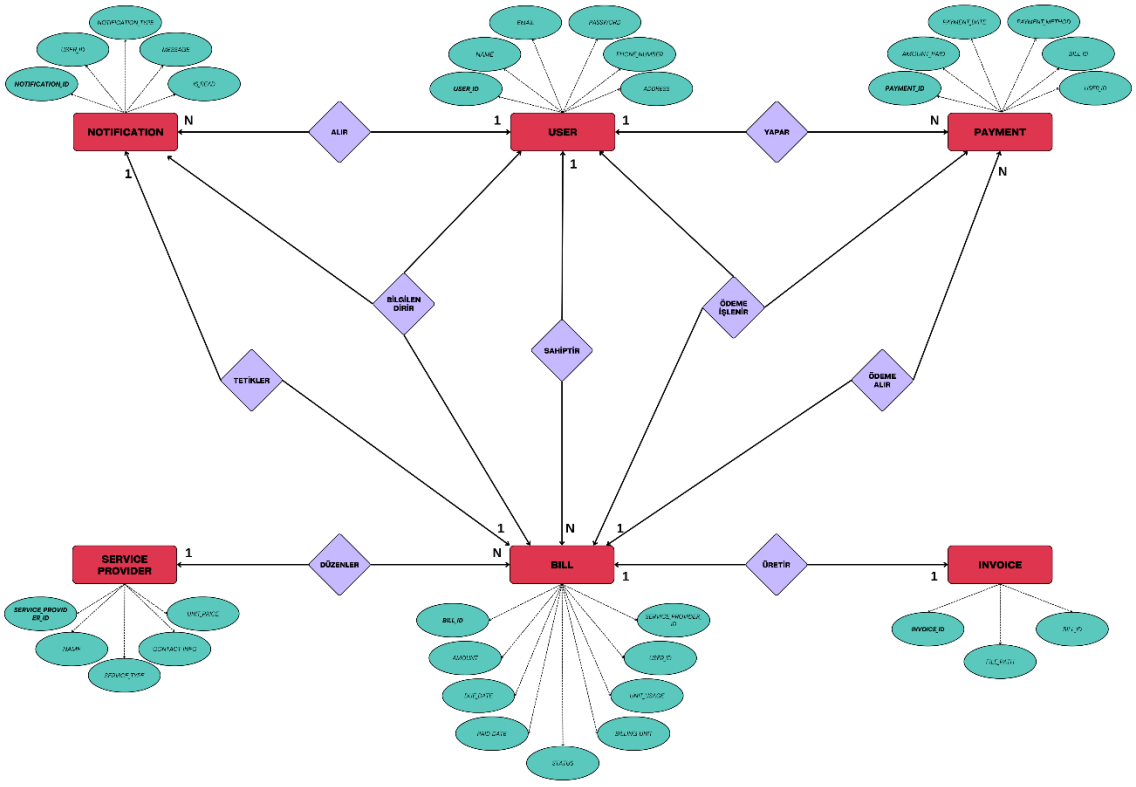
2. Servis Sağlayıcı Gereksinimleri

Fatura oluşturma ve güncelleme: Servis sağlayıcılar, müşterilere yeni faturalar ekleyebilir veya mevcutları düzenleyebilir.
Müşteri listesi ve detaylarını görüntüleme: Servis sağlayıcılar, hizmet verdikleri müşterilerin bilgilerine erişebilmelidir.
Bildirim gönderme: Servis sağlayıcılar, müşterilere toplu veya bireysel bildirimler gönderebilmelidir.

3. Yönetici Gereksinimleri

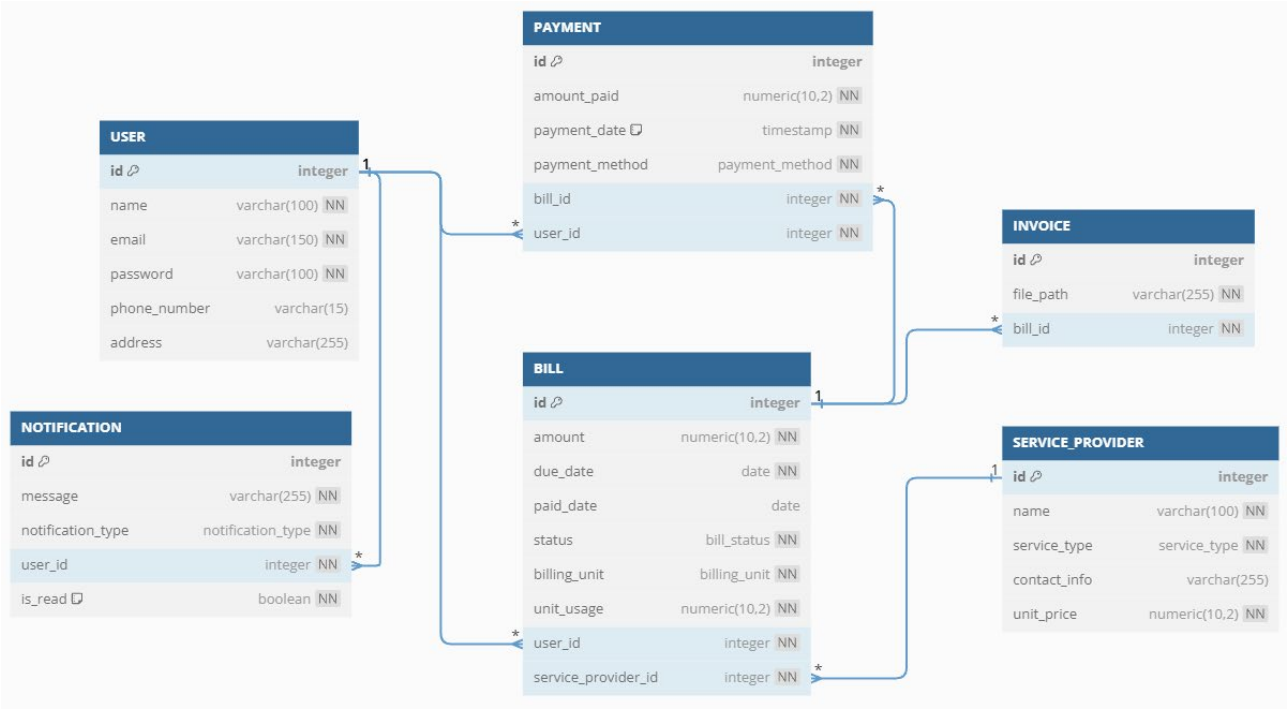
Kullanıcı yönetimi: Yöneticiler, kullanıcı hesaplarını oluşturabilir, düzenleyebilir veya silebilir.
Servis sağlayıcı yönetimi: Yöneticiler, yeni servis sağlayıcı ekleyebilir, düzenleyebilir veya kaldırabilir.
Bildirim oluşturma ve gönderme: Yöneticiler, kullanıcıları bilgilendirmek için genel bildirimler oluşturup gönderebilmelidir.

6. ER DİYAGRAMI



7. İLİŞKİSEL VERİ MODELİ

- USER(**ID**, NAME, EMAIL, PASSWORD, PHONE_NUMBER, ADDRESS)
- BILL(**ID**, AMOUNT, DUE_DATE, PAID_DATE, STATUS, BILLING_UNIT, UNIT_USAGE, USER_ID, SERVICE_PROVIDER_ID)
- PAYMENT(**ID**, AMOUNT_PAID, PAYMENT_DATE, PAYMENT_METHOD, BILL_ID, USER_ID)
- SERVICE_PROVIDER(**ID**, NAME, SERVICE_TYPE, CONTACT_INFO, UNIT_PRICE)
- NOTIFICATION(**ID**, USER_ID, NOTIFICATION_TYPE, MESSAGE, IS_READ)
- INVOICE(**ID**, FILE_PATH, BILL_ID)



8. TABLO NORMALLEŞTİRME

Veri tabanımızı normalleştirmeye çalıştırdığımızda BCNF sistemine dönüştürebildiğimizi fark ettik. BCNF (Boyce-Codd Normal Form) seviyesinin kuralı "Her belirleyen (bir kolon başka bir kolonun değerine ulaşabiliyorsa belirleyendir) bir aday anahtar olmalıdır." Bu kurala ulaşmak için tablo yapılarımızı dikkatlice analiz ettik ve aşağıdaki adımları izledik:

1. Kullanıcı ve Servis Sağlayıcı İlişkisi

Mevcut Sorun:

USER ve SERVICE_PROVIDER bağımlılığı:

- Bir kullanıcı birden fazla servis sağlayıcı ile ilişkilendirilebiliyordu, ancak bu ilişki BILL tablosunda tutuluyordu. Bu durum, user_id'nin service_provider_id'yi etkilediği, ancak user_id'nin bir aday anahtar olmadığı bir yapıya yol açıyordu.

Çözüm:

Yeni Tablo: USER_SERVICE_PROVIDER

- USER_SERVICE_PROVIDER tablosu ile kullanıcı ve servis sağlayıcı arasındaki ilişki izole edilmiştir.
- Bu iki alan (user_id, service_provider_id) birlikte birincil anahtar olarak tanımlanmıştır.
- Değerlendirme: Bu düzenleme doğru ve BCNF kuralına uygundur. USER_SERVICE_PROVIDER tablosu bağımlılıkları izole ederek diğer tabloları temizlemiştir.

USER TABLOSU	
user_id	name
1	Ali
2	Ayşe

SERVICE_PROVIDER TABLOSU	
service_provider_id	name
101	Elektrik AŞ
102	Su AŞ

BILL TABLOSU			
bill_id	amount	user_id	service_provider_id
1	100.00	1	101
2	50.0	1	102
3	200.0	2	101

USER_SERVICE_PROVIDER TABLOSU	
user_id	service_provider_id
1	101
1	102
2	101

2. BILL (Fatura) ve Servis Sağlayıcı (SERVICE_PROVIDER) İlişkisi

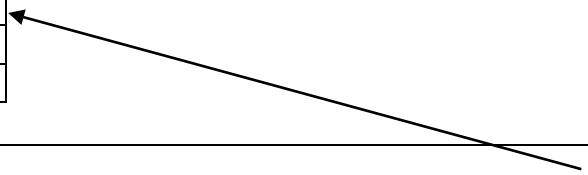
Mevcut Sorun:

- BILL tablosu içerisinde service_provider_id bilgisi doğrudan yer alıyordu, bu da kullanıcının birden fazla servis sağlayıcıyla ilişkilendirilmesinde transitif bağımlılıklara neden oluyordu.

Çözüm:

- BILL tablosundan service_provider_id kaldırılmıştır.
- Kullanıcıya bağlı bir faturanın ilgili servis sağlayıcı bilgisi artık USER_SERVICE_PROVIDER tablosu üzerinden takip edilebilmektedir.
- Değerlendirme: Bu düzenleme doğru. Transitif bağımlılıklar kaldırılarak BCNF'e uygun hale getirilmiştir.

BILL TABLOSU		
bill_id	amount	user_id
1	100.00	101
2	50.0	102



PAYMENT TABLOSU				
payment_id	amount_paid	payment_date	bill_id	user_id
1	100.00	2024-01-01	1	101
2	50.0	2024-01-15	1	101
3	200.0	2024-02-01	2	102

3. Fatura (BILL) ve Ödeme (PAYMENT) İlişkisi

Mevcut Sorun:

- PAYMENT tablosunda user_id bilgisi doğrudan yer alıyordu. Ancak, bu bilgi zaten bill_id üzerinden dolaylı olarak ilişkilendirilebiliyordu. Bu durum transitif bağımlılığa neden oluyordu.

Çözüm:

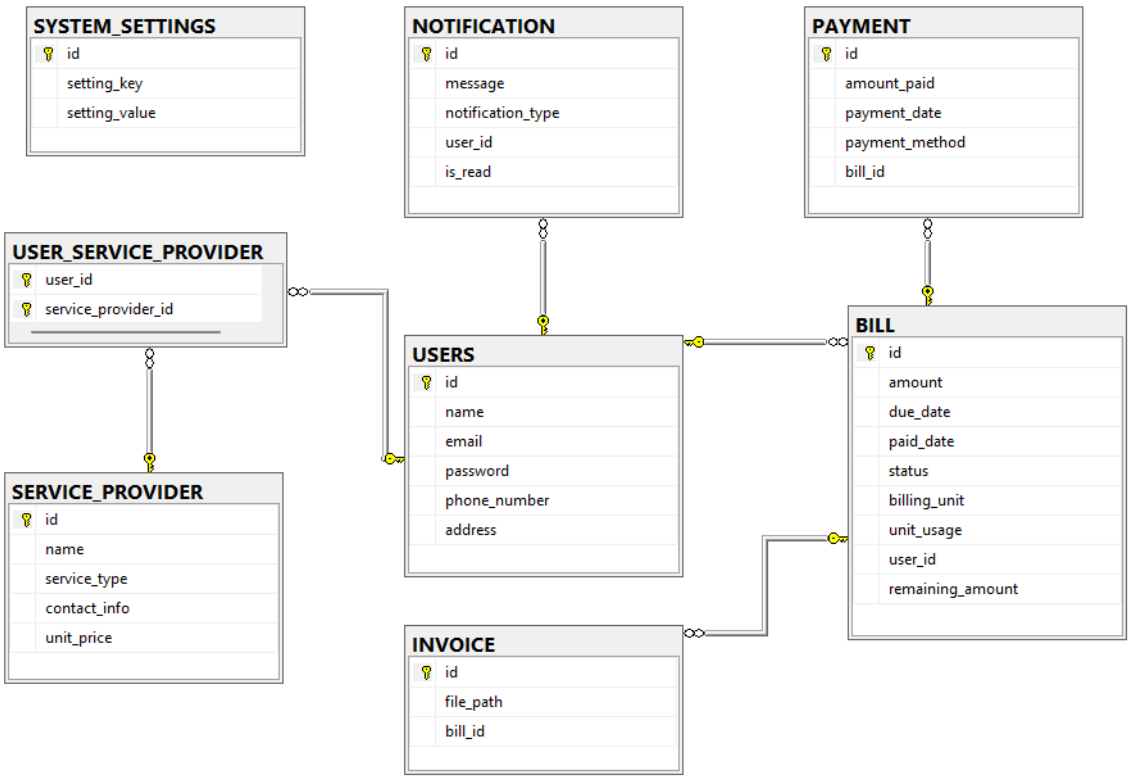
- PAYMENT tablosundan user_id kaldırılmıştır.
- Artık her ödeme doğrudan bir faturaya (bill_id) bağlanmıştır.
- Değerlendirme: Bu düzenleme doğru. Transitif bağımlılıklar ortadan kaldırılmıştır.

Sonuç

Yapılan normalleştirme adımları genel olarak BCNF seviyesini sağlamaktadır:

1. Her determinant bir aday anahtar olmuştur.
2. Tüm transitif bağımlılıklar kaldırılmıştır.
3. Veri tekrarları önlenmiş ve veri tabanı sorguları optimize edilmiştir.
4. Yapının genişletilebilirliği sağlanmıştır.

Bu normalizasyonlar sonucu elde ettiğimiz BCNF seviyesine ulaşmış veri tabanı ilişkisel diyagramımız aşağıdaki görseldeki gibi düzenlenmiştir:



9. SQL Komutlarıyla Şemanın Gerçekleştirimi:

```
-- USERS Tablosu
CREATE TABLE USERS (
    id INT PRIMARY KEY IDENTITY(1,1),
    name NVARCHAR(100) NOT NULL,
    email NVARCHAR(150) NOT NULL UNIQUE,
    password NVARCHAR(100) NOT NULL,
    phone_number NVARCHAR(15),
    address NVARCHAR(255)
);

-- SERVICE_PROVIDER Tablosu
CREATE TABLE SERVICE_PROVIDER (
    id INT PRIMARY KEY IDENTITY(1,1),
    name NVARCHAR(100) NOT NULL,
    service_type NVARCHAR(50) NOT NULL CHECK (service_type IN ('Elektrik', 'Su', 'İnternet')),
    contact_info NVARCHAR(255),
    unit_price DECIMAL(10,2) NOT NULL
);

-- USER_SERVICE_PROVIDER Tablosu
CREATE TABLE USER_SERVICE_PROVIDER (
    user_id INT NOT NULL,
    service_provider_id INT NOT NULL,
    PRIMARY KEY (user_id, service_provider_id),
    FOREIGN KEY (user_id) REFERENCES USERS (id) ON DELETE CASCADE,
    FOREIGN KEY (service_provider_id) REFERENCES SERVICE_PROVIDER (id) ON DELETE CASCADE
);
```

```
-- BILL Tablosu
CREATE TABLE BILL (
    id INT PRIMARY KEY IDENTITY(1,1),
    amount DECIMAL(10,2) NOT NULL,
    due_date DATE NOT NULL,
    paid_date DATE,
    status NVARCHAR(50) NOT NULL CHECK (status IN ('Ödenmiş', 'Ödenmemiş', 'Kısmi Ödenmiş')),
    billing_unit NVARCHAR(50) NOT NULL CHECK (billing_unit IN ('kWh', 'm³', 'GB')),
    unit_usage DECIMAL(10,2) NOT NULL,
    user_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES USERS (id) ON DELETE CASCADE
);

-- PAYMENT Tablosu
CREATE TABLE PAYMENT (
    id INT PRIMARY KEY IDENTITY(1,1),
    amount_paid DECIMAL(10,2) NOT NULL,
    payment_date DATETIME NOT NULL DEFAULT GETDATE(),
    payment_method NVARCHAR(50) NOT NULL CHECK (payment_method IN ('Kredi Kartı', 'Banka Transferi',
'Nakit')),
    bill_id INT NOT NULL,
    FOREIGN KEY (bill_id) REFERENCES BILL (id) ON DELETE CASCADE
);

-- NOTIFICATION Tablosu
CREATE TABLE NOTIFICATION (
    id INT PRIMARY KEY IDENTITY(1,1),
    message NVARCHAR(255) NOT NULL,
    notification_type NVARCHAR(50) NOT NULL CHECK (notification_type IN ('SMS', 'Email', 'Uygulama
Bildirimi')),
    user_id INT NOT NULL,
    is_read BIT NOT NULL DEFAULT 0,
    FOREIGN KEY (user_id) REFERENCES USERS (id) ON DELETE CASCADE
);

-- INVOICE Tablosu
CREATE TABLE INVOICE (
    id INT PRIMARY KEY IDENTITY(1,1),
    file_path NVARCHAR(255) NOT NULL,
    bill_id INT NOT NULL,
    FOREIGN KEY (bill_id) REFERENCES BILL (id) ON DELETE CASCADE
);
```

10. Tablolara Örnek Veriler Eklemek

- **USERS tablosu için örnek eklenen veriler**

```
INSERT INTO USERS (name, email, password, phone_number, address)
VALUES
('Ali Veli', 'ali@ornek.com', '******', '555-111-1111', 'İstanbul, Türkiye'),
('Ayşe Yılmaz', 'ayse@ornek.com', '******', '555-222-2222', 'Ankara, Türkiye'),
('ServisAdmin', 'admin@servis.com', '******', '555-333-3333', 'İzmir, Türkiye'),
('Yönetici', 'yonetici@ornek.com', '******', '555-444-4444', 'Antalya, Türkiye');
```

	id	name	email	password	phone_number	address
	1	Ali Veli	ali@ornek.com	******	555-111-1111	İstanbul, Türkiye
	2	Ayşe Yılmaz	ayse@ornek.com	******	555-222-2222	Ankara, Türkiye
	3	ServisAdmin	admin@servis.c...	******	555-333-3333	İzmir, Türkiye
	4	Yönetici	yonetici@ornek...	******	555-444-4444	Antalya, Türkiye
▶*	NULL	NULL	NULL	NULL	NULL	NULL

- **SERVICE_PROVIDER tablosu için örnek eklenen veriler**

```
INSERT INTO SERVICE_PROVIDER (name, service_type, contact_info, unit_price)
VALUES
('Elektrik AŞ', 'Elektrik', 'elektrik@servis.com', 1.2),
('Su Dağıtım AŞ', 'Su', 'su@servis.com', 0.9),
('İnternet AŞ', 'İnternet', 'internet@servis.com', 50.0);
```

	id	name	service_type	contact_info	unit_price
	1	Elektrik AŞ	Elektrik	elektrik@servis....	1,20
	2	Su Dağıtım AŞ	Su	su@servis.com	0,90
	3	İnternet AŞ	İnternet	internet@servis...	50,00
▶*	NULL	NULL	NULL	NULL	NULL

- **USER_SERVICE_PROVIDER tablosu için örnek eklenen veriler**

```
INSERT INTO USER_SERVICE_PROVIDER (user_id, service_provider_id)
VALUES
(1, 1),  -- Kullanıcı 1, Elektrik AŞ
(1, 2),  -- Kullanıcı 1, Su Dağıtım AŞ
(2, 3),  -- Kullanıcı 2, İnternet AŞ
(2, 2);  -- Kullanıcı 2, Su AŞ
```

	user_id	service_provid...
▶	1	1
	1	2
	2	2
	2	3
*	NULL	NULL

- **BILL tablosu için örnek eklenen veriler**

```
INSERT INTO BILL (amount, due_date, paid_date, status, billing_unit, unit_usage, user_id)
VALUES
(100.0, '2024-01-01', '2024-01-05', 'Ödenmiş', 'kWh', 80, 1),
(50.0, '2024-02-01', NULL, 'Ödenmemiş', 'm³', 10, 1),
(150.0, '2024-01-10', NULL, 'Kısmi Ödenmiş', 'GB', 30, 2);
```

	id	amount	due_date	paid_date	status	billing_unit	unit_usage	user_id
	1	100,00	2024-01-01	2024-01-05	Ödenmiş	kWh	80,00	1
	2	50,00	2024-02-01	*NULL*	Ödenmemiş	m^3	10,00	1
	3	150,00	2024-01-10	*NULL*	Kısmi Ödenmiş	GB	30,00	2
▶*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

- **PAYMENT tablosu için örnek eklenen veriler**

```
INSERT INTO PAYMENT (amount_paid, payment_date, payment_method, bill_id)
VALUES
(100.0, '2024-01-05', 'Kredi Kartı', 1),
(50.0, '2024-02-01', 'Banka Transferi', 2),
(50.0, '2024-01-10', 'Nakit', 3);
```

	id	amount_paid	payment_date	payment_met...	bill_id
	1	100,00	2024-01-05 00:0...	Kredi Kartı	1
	2	50,00	2024-02-01 00:0...	Banka Transferi	2
	3	50,00	2024-01-10 00:0...	Nakit	3
▶*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

- **NOTIFICATION tablosu için örnek eklenen veriler**

```
INSERT INTO NOTIFICATION (message, notification_type, user_id, is_read)
VALUES
('Faturanız ödenmiştir.', 'Email', 1, 1),
('Son ödeme tarihi yaklaşıyor.', 'SMS', 2, 0),
('Kısmi ödeme yapıldı.', 'Uygulama Bildirimi', 2, 1);
```

id	message	notification_type	user_id	is_read
1	Faturanız ödenmiştir.	Email	1	True
2	Son ödeme tarihi yaklaşıyor.	SMS	2	False
3	Kısmi ödeme yapıldı.	Uygulama Bildirimi	2	True
NULL	*NULL*	*NULL*	*NULL*	*NULL*

- **INVOICE tablosu için örnek eklenen veriler**

```
INSERT INTO INVOICE (file_path, bill_id)
VALUES
('/invoices/invoice1.pdf', 1),
('/invoices/invoice2.pdf', 2),
('/invoices/invoice3.pdf', 3);
```

	id	file_path	bill_id
	1	/invoices/invoice1.pdf	1
	2	/invoices/invoice2.pdf	2
	3	/invoices/invoice3.pdf	3
▶*	*NULL*	*NULL*	*NULL*

11. Gereksinimlere Göre Hazırlanmış Scriptler

a. Müşteri Gereksinimleri

1. Bir Müşterinin Faturayı Ödemesi;

```
-- Bunun için stored procedure olarak Process Payment adında programlanabilir bir script
yazdık. Aşağıdaki query ifadesinde olduğu gibi id'si 14 olan fatura(bill) in amount_paid
tutarndaki ödemesinin payment_method belirtilerek gerçekleştirildiğini görüyoruz.

EXEC ProcessPayment @bill_id = 14, @amount_paid = 50.0, @payment_method = 'Kredi Kartı';
```

NOT: Bir sonraki aşamada kullandığımız stored procedure olan ProcessPayment tanıtılacak.

2. Müşterinin Ödeme Geçmişini Görüntüleme;

```
DECLARE @user_id INT = 1;

-- Kullanıcının ödeme yaptığı faturaları ve ödeme detaylarını listeleme
SELECT
    p.id AS payment_id,
    p.amount_paid,
    p.payment_date,
    p.payment_method,
    b.id AS bill_id,
    b.amount AS bill_amount,
    b.due_date,
    b.paid_date,
    b.status
FROM PAYMENT p
INNER JOIN BILL b ON p.bill_id = b.id
WHERE b.user_id = @user_id
ORDER BY p.payment_date DESC;
```

3. Müşterinin Faturayı Görüntülemesi;

```
-- Faturalarını görüntüleyen kullanıcı için parametre
DECLARE @user_id INT = 1;

-- Kullanıcının faturalarını listele
SELECT
    b.id AS bill_id,
    b.amount,
    b.due_date,
    b.status,
    b.remaining_amount,
    sp.name AS service_provider_name
FROM BILL b
INNER JOIN SERVICE_PROVIDER sp ON b.service_provider_id = sp.id
WHERE b.user_id = @user_id;
```

4. Müşterinin Bildirim Geçmişini Görme

```
-- Bildirimleri görüntüleyecek kullanıcı için parametre
DECLARE @user_id INT = 2;

-- Kullanıcının bildirimlerini listele
SELECT n.id AS notification_id,
       n.message,
       n.notification_type,
       CASE
           WHEN n.is_read = 1 THEN 'Okundu'
           ELSE 'Okunmadı'
       END AS read_status,
       n.is_read,
       n.user_id
FROM NOTIFICATION n
WHERE n.user_id = @user_id
ORDER BY n.is_read ASC, n.id DESC; -- Önce okunmamış bildirimler
```

5. Müşterinin Ödemesini Yaptığı Fişleri Görüntülemesi

```
DECLARE @user_id INT = 1;

-- Kullanıcının ödenmiş faturalarına ait belgeleri listele
SELECT b.id AS bill_id,
       b.amount,
       b.due_date,
       b.paid_date,
       b.status,
       i.file_path AS invoice_path
FROM BILL b
INNER JOIN INVOICE i ON b.id = i.bill_id
WHERE b.user_id = @user_id
  AND b.status = 'Ödenmiş'
ORDER BY b.paid_date DESC;
```

b. Servis Sağlaycı Gereksinimleri

1. Servis Sağlayıcının Bir Müşteriye Ait Fatura Oluşturması

```
DECLARE @service_type NVARCHAR(50) = 'Su'; -- Seçilecek hizmet türü (Elektrik, Su,
İnternet)
DECLARE @user_id INT = 2; -- Kullanıcı ID
DECLARE @unit_usage DECIMAL(10,2) = 150.0; -- Kullanım miktarı
DECLARE @due_date DATE = DATEADD(DAY, 30, GETDATE()); -- Son ödeme tarihi
DECLARE @status NVARCHAR(50) = 'Ödenmemiş'; -- Varsayılan durum
DECLARE @billing_unit NVARCHAR(50); -- Hizmet türüne göre belirlenecek birim

-- Hizmet türüne göre billing_unit belirleme
SELECT TOP 1 @billing_unit =
    CASE
        WHEN @service_type = 'Elektrik' THEN 'kWh'
        WHEN @service_type = 'Su' THEN 'm³'
        WHEN @service_type = 'İnternet' THEN 'GB'
        ELSE NULL
    END;

IF @billing_unit IS NULL
BEGIN
    RAISERROR('Geçersiz hizmet türü.', 16, 1);
    RETURN;
END;

-- Fatura oluşturma
INSERT INTO BILL (amount, due_date, status, billing_unit, unit_usage, user_id,
remaining_amount)
SELECT
    sp.unit_price * @unit_usage AS amount, -- Tutar hesaplanıyor
    @due_date AS due_date,
    @status AS status,
    @billing_unit AS billing_unit,
    @unit_usage AS unit_usage,
    @user_id AS user_id,
    sp.unit_price * @unit_usage AS remaining_amount -- Kalan tutar eşitleniyor
FROM USER_SERVICE_PROVIDER usp
JOIN SERVICE_PROVIDER sp ON usp.service_provider_id = sp.id
WHERE usp.user_id = @user_id
  AND sp.service_type = @service_type; -- Dinamik hizmet türü seçimi
```

2. Servis Sağlayıcının Müşteriye Ait Bir Faturanın Güncellenmesi

```
DECLARE @bill_id INT = 1; -- Güncellenecek fatura ID'si
DECLARE @new_due_date DATE = '2024-03-15';

UPDATE BILL
SET due_date = @new_due_date
WHERE id = @bill_id;
```

3. Servis Sağlayıcının Müşterilerine Ait Fatura Detaylarının Görüntülenmesi

```
DECLARE @service_provider_id INT = 3;

SELECT b.id AS bill_id,
       b.amount,
       b.due_date,
       b.paid_date,
       b.status,
       b.billing_unit,
       b.unit_usage,
       u.name AS customer_name,
       u.email
FROM BILL b
JOIN USERS u ON b.user_id = u.id
JOIN USER_SERVICE_PROVIDER usp ON u.id = usp.user_id
WHERE usp.service_provider_id = @service_provider_id
ORDER BY b.due_date ASC;
```

4. Servis Sağlayıcının Müşterilerine Toplu Bildirim Göndermesi

```
DECLARE @message NVARCHAR(255) = 'Yeni fiyat tarifesi bilgisi!';
DECLARE @notification_type NVARCHAR(50) = 'Email';
DECLARE @service_provider_id INT = 2;

INSERT INTO NOTIFICATION (message, notification_type, user_id, is_read)
SELECT @message, @notification_type, usp.user_id, 0
FROM USER_SERVICE_PROVIDER usp
WHERE usp.service_provider_id = @service_provider_id;
```

5. Bir Servis Sağlayıcının Kendisine Bağlı Müşterilerini Görmesi

```
DECLARE @service_provider_id INT = 2;

SELECT u.id AS customer_id,
       u.name AS customer_name,
       u.email,
       u.phone_number
FROM USERS u
JOIN USER_SERVICE_PROVIDER usp ON u.id = usp.user_id
WHERE usp.service_provider_id = @service_provider_id
ORDER BY u.name ASC;
```

6. Bir Servis Sağlayıcının Hizmet Birim Fiyatını Güncellemesi

```
DECLARE @service_provider_id INT = 1;
DECLARE @new_unit_price DECIMAL(10,2) = 1.5;

UPDATE SERVICE_PROVIDER
SET unit_price = @new_unit_price
WHERE id = @service_provider_id;
```

7. Bir Servis Sağlayıcının Müşterilerinin Ödeme Durumlarını Takip Etmesi

```
DECLARE @service_provider_id INT = 2;

SELECT b.id AS bill_id,
       b.amount,
       b.due_date,
       b.paid_date,
       b.status,
       u.name AS customer_name,
       u.email
FROM BILL b
JOIN USERS u ON b.user_id = u.id
JOIN USER_SERVICE_PROVIDER usp ON u.id = usp.user_id
WHERE usp.service_provider_id = @service_provider_id
ORDER BY b.status ASC, b.due_date ASC;

-- Eğer sadece bir müşterisinin durumunu takip etmek isterse

DECLARE @service_provider_id INT = 1; -- Servis sağlayıcının ID'si
DECLARE @user_id INT = 1; -- Kullanıcının ID'si

SELECT b.id AS bill_id,
       b.amount,
       b.due_date,
       b.paid_date,
       b.status,
       b.billing_unit,
       b.unit_usage
FROM BILL b
JOIN USERS u ON b.user_id = u.id
JOIN USER_SERVICE_PROVIDER usp ON u.id = usp.user_id
WHERE usp.service_provider_id = @service_provider_id
  AND u.id = @user_id
ORDER BY b.due_date ASC;
```

c. Yönetici Gereksinimleri

1. Yeni Kullanıcı Oluşturma

```
DECLARE @name NVARCHAR(100) = 'Zehra Çelik';
DECLARE @email NVARCHAR(150) = 'zehra@ornek.com';
DECLARE @password NVARCHAR(100) = '******';
DECLARE @phone_number NVARCHAR(15) = '555-555-5555';
DECLARE @address NVARCHAR(255) = 'İstanbul, Türkiye';

INSERT INTO USERS (name, email, password, phone_number, address)
VALUES (@name, @email, @password, @phone_number, @address);
```

2. Bir Kullanıcıyı Güncelleme

```
DECLARE @user_id INT = 1;
DECLARE @new_email NVARCHAR(150) = 'ali.veli@ornek.com';
DECLARE @new_phone NVARCHAR(15) = '555-666-7777';
DECLARE @new_address NVARCHAR(255) = 'Ankara, Türkiye';

UPDATE USERS
SET email = @new_email,
    phone_number = @new_phone,
    address = @new_address
WHERE id = @user_id;
```

3. Bir Kullanıcıyı Silme

```
DECLARE @user_id INT = 6;

DELETE FROM USERS
WHERE id = @user_id;
```

4. Yeni Servis Sağlayıcı Eklenmesi

```
DECLARE @name NVARCHAR(100) = 'Marsu AŞ';
DECLARE @service_type NVARCHAR(50) = 'Su';
DECLARE @contact_info NVARCHAR(255) = 'marsu@servis.com';
DECLARE @unit_price DECIMAL(10,2) = 2.5;

INSERT INTO SERVICE_PROVIDER (name, service_type, contact_info, unit_price)
VALUES (@name, @service_type, @contact_info, @unit_price)
```

5. Bir Servis Sağlayıcının Güncellenmesi

```
DECLARE @service_provider_id INT = 5;
DECLARE @new_unit_price DECIMAL(10,2) = 1.5;

UPDATE SERVICE_PROVIDER
SET unit_price = @new_unit_price
WHERE id = @service_provider_id;
```

6. Bir Servis Sağlayıcının Silinmesi

```
DECLARE @service_provider_id INT = 5;

DELETE FROM SERVICE_PROVIDER
WHERE id = @service_provider_id;
```

7. Sistem Ayarları Eklemek

```
-- Sistem ayarları tablosu oluşturma
CREATE TABLE SYSTEM_SETTINGS (
    id INT PRIMARY KEY IDENTITY(1,1),
    setting_key NVARCHAR(100) UNIQUE NOT NULL,
    setting_value NVARCHAR(255) NOT NULL
);
-- Yeni bir sistem ayarı ekleme
DECLARE @setting_key NVARCHAR(100) = 'notification_enabled';
DECLARE @setting_value NVARCHAR(255) = 'true';

INSERT INTO SYSTEM_SETTINGS (setting_key, setting_value)
VALUES (@setting_key, @setting_value);
```

8. Sistem Ayarını Güncellemek

```
-- Mevcut bir ayarı güncelleme
DECLARE @setting_key NVARCHAR(100) = 'notification_enabled';
DECLARE @new_setting_value NVARCHAR(255) = 'false';

UPDATE SYSTEM_SETTINGS
SET setting_value = @new_setting_value
WHERE setting_key = @setting_key;
```

9. Tüm Kullanıcılara Genel Bildirim Gönderilmesi

```
DECLARE @message NVARCHAR(255) = 'Sistem bakımı yapılacaktır.';
DECLARE @notification_type NVARCHAR(50) = 'Email';

INSERT INTO NOTIFICATION (message, notification_type, user_id, is_read)
SELECT @message, @notification_type, u.id, 0
FROM USERS u;
```

10. Belirli Bir Kullanıcı Grubuna Bildirim Gönderimi

```
-- Belirli bir gruba (örneğin, belirli bir servis sağlayıcıya bağlı olan kullanıcılar)
bildirim gönderimi
DECLARE @message NVARCHAR(255) = 'Yeni fiyat tarifesi bilgisi!';
DECLARE @notification_type NVARCHAR(50) = 'SMS';
DECLARE @service_provider_id INT = 1;

INSERT INTO NOTIFICATION (message, notification_type, user_id, is_read)
SELECT @message, @notification_type, usp.user_id, 0
FROM USER_SERVICE_PROVIDER usp
WHERE usp.service_provider_id = @service_provider_id;
```

12. Saklı Yordam ve Tetikleyici Kullanımı (Stored Procedures- Triggers)

1. Saklı Yordam – Process Payment:

Projede bir adet saklı yordam kullandık. Bu saklı yordam, bir faturanın ödenmesi işlemini yönetmek için tasarlanmıştır. Ödemenin doğruluğunu kontrol eder, ödemeyi kaydeder ve fatura durumunu günceller. Detaylar;

Parametreler

@bill_id (INT): Ödeme yapılacak fatura ID'si.

@amount_paid (DECIMAL(10,2)): Yapılan ödeme miktarı.

@payment_method (NVARCHAR(50)): Kullanılan ödeme yöntemi (örn. Kredi Kartı, Nakit).

Transaction Başlatma:

Tüm işlemler bir transaction içinde yürütülür. Bu sayede hata durumunda işlemler geri alınabilir.

Kontroller

- bill_id kontrol edilir olmaması durumunda hata handle edilir.
- Fazla ödeme kontrolü yapılır bu sayede hataların önüne geçilir.
- Ödeme bilgileri Payment tablosuna kaydedilir.
- Eğer kısmi ödeme yapılmışsa BILL tablosunda remaining_amount (kalan tutar) kolonu güncellenir.
- Faturanın status(durum) durumu kısmi ödenmiş veya ödenmiş olarak değiştirilir. Eğer tamamı ödenmişse paid_date tarihi atanır.
- Rollback Transaction ile hata durumları handle edilir ve hata detay mesajları döndürülür.

dbo.ProcessPayment:

```
USE [fys_1]
GO
/****** Object:  StoredProcedure [dbo].[ProcessPayment]    Script Date: 21.12.2024 16:12:51 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[ProcessPayment]
    @bill_id INT,
    @amount_paid DECIMAL(10,2),
    @payment_method NVARCHAR(50)
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION;

        -- Fatura ID'nin varlığını kontrol et
        IF NOT EXISTS (SELECT 1 FROM BILL WHERE id = @bill_id)
        BEGIN
            RAISERROR('Geçersiz fatura ID.', 16, 1);
            RETURN;
        END

        -- Fazla ödeme kontrolü
        DECLARE @total_amount DECIMAL(10,2);
        DECLARE @remaining_amount DECIMAL(10,2);

        SELECT @total_amount = amount, @remaining_amount = remaining_amount
        FROM BILL
        WHERE id = @bill_id;

        IF @remaining_amount < @amount_paid
        BEGIN
                        PRINT 'Kalan tutar: ' + STR(@remaining_amount, 10, 2);
                        RAISERROR('Ödeme miktarı kalan tutarı aşamaz. Lütfen kalan tutarı kontrol edin.', 16,
1);
            ROLLBACK TRANSACTION;
            RETURN;
        END

        -- Ödeme bilgilerini ekle
        INSERT INTO PAYMENT (amount_paid, payment_date, payment_method, bill_id)
        VALUES (@amount_paid, GETDATE(), @payment_method, @bill_id);

        -- Kalan tutarı güncelle
        SET @remaining_amount = @remaining_amount - @amount_paid;

        UPDATE BILL
        SET remaining_amount = @remaining_amount
        WHERE id = @bill_id;

        -- Fatura durumu güncelle
        IF @remaining_amount = 0
        BEGIN
            UPDATE BILL
            SET status = 'Ödenmiş', paid_date = GETDATE()
            WHERE id = @bill_id;
        END
        ELSE
        BEGIN
            UPDATE BILL
            SET status = 'Kısmi Ödenmiş'
            WHERE id = @bill_id;
        END

        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Hata durumunda işlemi geri al ve hata mesajını döndür
        ROLLBACK TRANSACTION;
        DECLARE @error_message NVARCHAR(4000) = ERROR_MESSAGE();
        RAISERROR('Hata: %s', 16, 1, @error_message);
    END CATCH;
END;
```

2. Tetikleyici – AfterBillUpdate

Bu tetikleyici, BILL tablosundaki bir güncelleme (UPDATE) işlemi sonrasında çalışan bir mantık sağlar. BILL tablosunda yapılan değişikliklere göre fatura durumunu ve kullanıcıya gönderilecek bildirimleri otomatik olarak yönetir.

Amaç

Fatura Durum Güncellemesi:

- remaining_amount alanına bağlı olarak faturanın durumunu (status) günceller.
- Kalan tutar sıfır olduğunda durum "Ödenmiş", aksi halde "Kısmi Ödenmiş" olarak belirlenir.

Bildirim Gönderimi:

- Kullanıcıya SMS şeklinde bildirim oluşturur ve NOTIFICATION tablosuna ekler.

Kullanılan Değişkenler

Fatura ve Kullanıcı Bilgileri:

- @bill_id: Güncellenen fatura ID'si.
- @new_status: Faturanın yeni durumu.
- @user_id: Fatura ile ilişkili kullanıcı ID'si.
- @remaining_amount: Faturanın kalan ödeme miktarı.

Bildirimle İlgili Bilgiler:

- @notification_message: Kullanıcıya gönderilecek bildirim mesajı.
- @notification_type: Bildirim türü (ör. "SMS").

İşleyiş Adımları

- Çoklu satır güncelleme için Cursor kullanımı ile birlikte tetikleyicinin aynı anda birden fazla satırın güncellenebileceği senaryolar için INSERTED sanal tablosunu cursor ile dolaşır. Güncellenen her satırdan id, status, user_id ve remaining_amount bilgilerini alır.
- Fatura durum kontrolü yapar. Kalan tutara göre status durumunu düzenler ve müşteriye notification gönderir.
- Müşterilere gönderilecek bildirimler notification tablosuna eklenir.
- Yani güncellenen her fatura için status ve notification veritabanında kaydedilir.
- Hata yönetimi yapılır ve hata mesajları kaydedilir.
- Açık cursor temizlenir kapatılır.

```
USE [fys_1]
GO
/****** Object:  Trigger [dbo].[AfterBillUpdate]    Script Date: 5.01.2025 16:21:48 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[AfterBillUpdate]
ON [dbo].[BILL]
AFTER UPDATE
AS
BEGIN
    BEGIN TRY
        -- Eğer cursor zaten açık ise kapat ve temizle
        IF CURSOR_STATUS('GLOBAL', 'cursor_inserted') >= 0
        BEGIN
            CLOSE cursor_inserted;
            DEALLOCATE cursor_inserted;
        END

        -- Çoklu satır güncelleme için cursor kullanımı
        DECLARE @bill_id INT;
        DECLARE @new_status NVARCHAR(50);
        DECLARE @user_id INT;
        DECLARE @remaining_amount DECIMAL(10,2);
        DECLARE @notification_message NVARCHAR(255);
        DECLARE @notification_type NVARCHAR(50);

        DECLARE cursor_inserted CURSOR FOR
        SELECT id, status, user_id, remaining_amount FROM inserted;

        OPEN cursor_inserted;

        FETCH NEXT FROM cursor_inserted INTO @bill_id, @new_status, @user_id, @remaining_amount;

        WHILE @@FETCH_STATUS = 0
        BEGIN
            -- Bildirim türü sabit olarak atanıyor
            SET @notification_type = 'SMS';

            -- Kalan tutara göre durum kontrolü
            IF @remaining_amount = 0
            BEGIN
                SET @new_status = 'Ödenmiş';
                SET @notification_message = CONCAT('Fatura ID ', @bill_id, ' başarıyla ödenmiştir. Teşekkür
ederiz!');
            END
            ELSE
            BEGIN
                SET @new_status = 'Kısmi Ödenmiş';
                SET @notification_message = CONCAT('Fatura ID ', @bill_id,
                                                   ' için kalan ödeme bulunmaktadır. Kalan tutar: ',
                                                  CAST(@remaining_amount AS NVARCHAR(50)), ' TL.');
            END
```

```
-- Durumu güncelle
            UPDATE BILL
            SET status = @new_status
            WHERE id = @bill_id;

            -- Bildirimi ekle
            INSERT INTO NOTIFICATION (message, notification_type, user_id, is_read)
            VALUES (@notification_message, @notification_type, @user_id, 0);

            FETCH NEXT FROM cursor_inserted INTO @bill_id, @new_status, @user_id, @remaining_amount;
        END;

        CLOSE cursor_inserted;
        DEALLOCATE cursor_inserted;
    END TRY
    BEGIN CATCH
        PRINT 'Tetikleyicide hata: ' + ERROR_MESSAGE();
        IF CURSOR_STATUS('GLOBAL', 'cursor_inserted') >= 0
        BEGIN
            CLOSE cursor_inserted;
            DEALLOCATE cursor_inserted;
        END
    END CATCH;
END;
```