

SAKARYA ÜNİVERSİTESİ BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ VERİTABANI YÖNETİM SİSTEMLERİ DERSİ PROJE
ÖDEVİ

HAZIRLAYAN ÖĞRENCİLER:

YASİN CAN KAYA G221210021

MELİKE DEMİRTAŞ G211210003

GRUP : 2-B

Tanıtım: Online alışveriş sisteminden, kullanıcıların bilgilerinin (ad, soyadı, adres, telefon numarası, adres vb.) güvenliğinin sağlanması, depolanması, satılacak tüm ürün bilgilerin depolanması, tüm satılan ürünlerin bilgilerinin saklanması istenmektedir.

İş Kuralları:

Her kullanıcı sadece kendine ait(benzersiz) bir kullanıcı koduna sahiptir. Kullanıcıların ad, soyadı, e-posta, şifre, telefon numarası, doğum tarihi bulunur.

Ürünlerin adı, fiyatı, stok bilgileri, üretim tarihi ve her ürün için birbirinden farklı ürün kodları bulunur.

Kategorilerin bilgileri içerisinde kategori kodu ve kategorilerin isimleri bulunur.

Sipariş bilgisi içinde sepet kodu, tarih, fiyat bulunur.

Sipariş ürünleri bilgisi içinde ürünlerin kodu, sipariş kodu, stok, bulunur.

Ödeme içinde ödeme kodu, fiyat, ödeme tarihi bilgileri bulunur.

Fatura içinde fatura kodu, fiyat ve fatura tarihi bilgileri bulunur.

Kargo firması içinde firma kodu, firma adı, telefon numarası yer alır.

Kargo bilgileri içinde takip numarası ve teslimat tarihi ve kargo kodu bulunur.

İade talepleri içinde talep kodu, talep tarihi ve durum bilgileri yer alır.

Geri Bildirimler içinde geri bildirim kodu, yorum, puan bilgileri bulunur.

Kartla ödeme içinde ödeme kodu ve taksit miktarı bilgileri yer alır.

Kapıda ödeme içinde ödeme kodu ve ekstra kargo ücreti bulunur.

Kullanıcı oturumu içinde oturum kodu, başlangıç zamanı ve bitiş zamanı yer alır.

Satıcı içinde satıcı kodu, satıcı adı, adres, telefon, e-posta bilgileri bulunur.

Her kullanıcı bir siparişe sahip olabilir de hiç siparişe sahip olmayabilir de ama her sipariş sadece bir kullanıcı tarafından sahip olunabilir.

Bir kategoriye ait birden çok ürün olabilir ama bir ürün sadece bir kategoriye aittir.

Sepetteki ürünler birden fazla olabilir her bir ürün de birden fazla sepette bulunabilir.

Her ödeme bir siparişe aittir ve her sipariş için sadece bir tane ödeme yapılabilir.

Her kargo bilgisi bir sipariş için olur ama her sipariş birden fazla kargo bilgisi içerebilir.

Her iade talebi bir siparişe aittir her iade talebi de bir sipariş için oluşturabilir ama hiç iade talebi de oluşturmayabilir.

Her iade talebi bir kullanıcıya aittir. Her kullanıcı da bir tane iade talebi oluşturur ya da hiç oluşturmayabilir

Her geri bildirim bir kullanıcıya aittir ama her kullanıcı birden fazla geri bildirimde bulunabilir. Hiç geri bildirimde bulunmayabilir de.

Her kupon bir siparişe ait olabilir ya da hiçbir siparişe ait olmayabilir, bir siparişte yalnız bir kupon kullanılabilir ya da hiç kullanılmayabilir.

Her oturum bir kullanıcıya aittir her kullanıcı birden fazla oturum açabilir.

Her satıcı birden fazla ürün satabilir her ürün de birden fazla satıcı tarafından satılabilir.

Her fatura bir siparişe aittir. Her siparişin bir faturası olur.

Bir kargo firmasının birden fazla kargo bilgisi olabilir. Bir kargo bilgisi bir kargo firmasına aittir.

İlişkisel Şema:

kullanici kullaniciID: string, ad: string, soyad: string, eposta: string, telefonNo: char (11)dogumTarihi: char)

urun(urunID: string, ad: string, fiyat: integer, uretimTarihi: char)

siparis(siparisID: string, fiyat: integer, tarih: char)

[illegible]

Form1

btnKullaniciEkle btnKullaniciSil btnKullaniciGuncelle btnKullaniciAra UrunFiyatArtir

btnSiparisEkle TarihGecenKuponlanSil

KullaniciID SonOturumBul

KullaniciID KullanciYasBul

Kullanici başarıyla eklendi.

Tamam

Dependencies Dependents Processes **sqlkayit.sql***

email.com', 'ahmet123', '1234567890', '1990-01-15'),
ail.com', 'ayse456', '9876543210', '1985-05-20');

ter varying (50) eposta character varying (50) sifre character varying (50) telefonno character varying (15) dogumtarihi date

2	7	asd			123123	31321	12312	2023-12-08
3	8		asd					2023-12-26
4	11	1231		sa				2023-12-26
5	10					asds		2023-12-26
6	12							2023-12-26
7	13	Ahmet	Yilmaz	ahmet@email.com	ahmet123	1234567890		1990-01-15
8	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210		1985-05-20
9	15	mehmet	ak	mehmet123@gmail.com	123	533888333122		2000-02-02

Triggers

- kupon
- odeme
- satıcı
- satıcıurun
- siparis
- siparisurun
- urun

Trigger Functions (4)

- trigooer siparis ekle onuncelle()

Kullanıcı Silme:

Form1

KullaniciEkle

Ad mehmehmet

Soyad alk

Eposta mehmehmet123@gmail.co

Sifre 123

Telefon 533888333122

DogumTarihi 2 Subat 2000 Cargamba

Onayla

UrunFiyatArtir

TarihGecenKuponlanSil

Kull

KullanciYasBul

Dependencies Dependents Processes **sqlkayit.sql***

email.com', 'ahmet123', '1234567890', '1990-01-15'),
ail.com', 'ayse456', '9876543210', '1985-05-20');

Fiyat, tarih ve kullanıcıID'yi uygun şekilde güncelleyin
Fiyat, tarih ve kullanıcıID'yi uygun şekilde güncelleyin
ad, eposta, sifre, telefonNo, dogumTarihi)

ter varying (50) eposta character varying (50) sifre character varying (50) telefonno character varying (15) dogumtarihi date

2	7	asd			123123	31321	12312	2023-12-08
3	8		asd					2023-12-26
4	11	1231		sa				2023-12-26
5	10					asds		2023-12-26
6	12							2023-12-26
7	13	Ahmet	Yilmaz	ahmet@email.com	ahmet123	1234567890		1990-01-15
8	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210		1985-05-20

Triggers

- kupon
- odeme
- satıcı
- satıcıurun
- siparis
- siparisurun
- urun

Trigger Functions (4)

Form1

KullaniciGuncelle

Ad: akif
Soyad: yilmaz
Eposta: akifyilmaz@gmail.com
Sifre: 3456
TelNo: 53555577723
DogumTarihi: 2 Şubat 2000 Çarşamba
KullaniciID: 15
Onayla

btnKullaniciEkle btnKullaniciSil btnKullaniciGuncelle btnKullaniciAra
btnSiparisEkle
UrunFiyatAtir
TarihGececiKuponlariSil

KullaniciID: SonOturumBul
KullaniciID: KullanciYasBul

id	id	id	id	id	id	id	id
1	7	asd					2023-12-26
2	8		asd				2023-12-26
3	11		1231	sa			2023-12-26
4	10				asds		2023-12-26
5	12						2023-12-26
6	13	Ahmet	Yilmaz	ahmet@email.com	ahmet123	1234567890	1990-01-15
7	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210	1985-05-20
8	15	mehmet	ak	mehmet123@gmail.com	123	533888333122	2000-02-02
9	16						2023-12-26

Güncelleme:

Form1

KullaniciGuncelle

Ad: akif
Soyad: yilmaz
Eposta: akifyilmaz@gmail.com
Sifre: 3456
TelNo: 53555577723
DogumTarihi: 2 Şubat 2000 Çarşamba
KullaniciID: 15
Onayla

btnKullaniciEkle btnKullaniciSil btnKullaniciGuncelle btnKullaniciAra
btnSiparisEkle
UrunFiyatAtir
TarihGececiKuponlariSil

KullaniciID: SonOturumBul
KullaniciID: KullanciYasBul

id	id	id	id	id	id	id	id
1	7	asd					2023-12-26
2	8		asd				2023-12-26
3	11		1231	sa			2023-12-26
4	10				asds		2023-12-26
5	12						2023-12-26
6	13	Ahmet	Yilmaz	ahmet@email.com	ahmet123	1234567890	1990-01-15
7	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210	1985-05-20
8	15	mehmet	ak	mehmet123@gmail.com	123	533888333122	2000-02-02
9	16						2023-12-26

Form1

Dashboard Properties S

postgres/postgres@P

Query Query History

```
355 -- ('Ahmet',
356 -- ('Ayşe',
357
358 select*from kulla
359
360
361
```

Data Output Messages

KullanıcıID

SonOturumBul

KullanıcıID

KullanıcıYaşBul

btnKullanıcıEkle

btnKullanıcıSil

btnKullanıcıGuncelle

btnKullanıcıAra

UrunFiyatArttir

TarihGecenKuponlarıSil

btnSiparisEkle

Kullanıcı başarıyla güncellendi.

Tamam

kullaniciid	ad	soyad	email	password	telefon	dogumtarihi	
1	7	asd				2023-12-26	
2	8					2023-12-26	
3	11					2023-12-26	
4	10			asds		2023-12-26	
5	12					2023-12-26	
6	13	Ahmet	Yılmaz	ahmet@email.com	ahmet123	1234567890	1990-01-15
7	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210	1985-05-20
8	16						2023-12-26
9	15	akif	yilmaz	akifyilmaz@gmail.com	3456	53555577723	2000-02-02

Arama:

Form1

Tools Help

KullanıcıAra

kullaniciid 15

Ara

btnKullanıcıEkle

btnKullanıcıSil

btnKullanıcıGuncelle

btnKullanıcıAra

UrunFiyatArttir

TarihGecenKuponlarıSil

btnSiparisEkle

KullanıcıID

SonOturumBul

KullanıcıID

KullanıcıYaşBul

Triggers

- kupon
- odeme
- satıcı
- satıcıurun
- siparis
- siparisurun
- urun

Trigger Functions (4)

- trigger_siparis_ekle_guncelle()

kullaniciid	ad	soyad	email	password	telefon	dogumtarihi	
2	8		asd			2023-12-26	
3	11		1231	sa		2023-12-26	
4	10				asds	2023-12-26	
5	12					2023-12-26	
6	13	Ahmet	Yılmaz	ahmet@email.com	ahmet123	1234567890	1990-01-15
7	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210	1985-05-20
8	16						2023-12-26
9	15	akif	yilmaz	akifyilmaz@gmail.com	3456	53555577723	2000-02-02

The screenshot shows a PostgreSQL database interface. On the left, a query window displays a SQL query: `select * from kullanicilar`. Below the query window, a data output table shows the results of the query. The table has columns: `kullaniciid` (integer), `ad` (character varying), `soyad` (character varying), `email` (character varying), `telefon` (character varying), and `tarih` (date). The table contains 9 rows of data.

In the center, a user interface form titled "Form1" is displayed. It contains several buttons: `btnKullaniciEkle`, `btnKullaniciSil`, `btnKullaniciGuncelle`, `btnKullaniciAra`, `btnSiparisEkle`, `UrunFiyatAttir`, `TarihGecenKuponlariSil`, `KullaniciID` (text input), `SonOturumBul` (button), `KullaniciYasiBul` (button), and `KullaniciID` (text input). A small dialog box is open over the form, displaying the following information: `Kullanici Bulundu:`, `ID: 15`, `Ad: akif`, `Soyad: yilmaz`, `Eposta: akifyilmaz@gmail.com`, `Telefon: 53555577723`. A `Tamam` button is at the bottom of the dialog.

	kullaniciid [PK] integer	ad character varying	soyad character varying	email character varying	telefon character varying	tarih date
1	7	asd				2023-12-26
2	8					2023-12-26
3	11					2023-12-26
4	10					2023-12-26
5	12					2023-12-26
6	13	Ahmet	Yilmaz	ahmet@email.com	ahmet123	1234567890
7	14	Ayşe	Demir	ayse@email.com	ayse456	9876543210
8	16					2023-12-26
9	15	akif	yilmaz	akifyilmaz@gmail.com	3456	53555577723

Fonksiyonlar:

- **CREATE OR REPLACE FUNCTION trigger_yeni_siparis_eklendiginde_odeme_ekle()**

-- RETURNS TRIGGER AS \$\$

-- BEGIN

-- -- Yeni bir sipariş eklenirse, yeni bir ödeme satırı ekleyin

-- INSERT INTO odeme (fiyat, odemeTarihi, kullaniciID)

-- VALUES (NEW.fiyat, CURRENT_DATE, NEW.kullaniciID);

-- RETURN NEW;

-- END;

-- \$\$ LANGUAGE plpgsql;

Açıklama: Tabloya yeni sipariş eklendiğinde ödenme tablosuna da ekler.

-- **CREATE OR REPLACE FUNCTION trigger_siparis_ekle_guncelle()**

-- RETURNS TRIGGER AS \$\$

-- BEGIN

-- -- Yeni bir sipariş eklendiğinde veya güncellendiğinde yapılacak işlemler buraya eklenir

-- IF TG_OP = 'INSERT' THEN

```

--      RAISE NOTICE 'Yeni sipariş eklendi: %', NEW.siparisID;
--      ELSEIF TG_OP = 'UPDATE' THEN
--      RAISE NOTICE 'Sipariş güncellendi: %', NEW.siparisID;
--      END IF;

--      RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;
-- CREATE TRIGGER siparis_trigger
-- AFTER INSERT OR UPDATE ON siparis
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_siparis_ekle_guncelle();

```

Açıklama: Yeni sipariş eklenince notice atıyor.

```

-- CREATE OR REPLACE FUNCTION trigger_yeni_kullanici_eklendiginde_oturum_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--      -- Yeni bir kullanıcı eklenirse, yeni bir oturum satırı ekleyin
--      INSERT INTO kullanıcıOturum (başlangicZamani, bitisZamani, kullanıcıID)
--      VALUES (CURRENT_TIMESTAMP, NULL, NEW.kullanıcıID);

--      RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;
-- CREATE TRIGGER yeni_kullanici_trigger
-- AFTER INSERT ON kullanıcı
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_yeni_kullanici_eklendiginde_oturum_ekle();

```

Açıklama: Yeni kullanıcı eklendiğinde oturum da ekleniyor.

```

-- CREATE OR REPLACE FUNCTION trigger_yeni_siparis_eklendiginde_odeme_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--      -- Yeni bir sipariş eklenirse, yeni bir ödeme satırı ekleyin
--      INSERT INTO ödeme (fiyat, ödemeTarihi, siparisID)
--      VALUES (NEW.fiyat, CURRENT_DATE, NEW.siparisID);

--      RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;

```



```
-- CREATE TRIGGER yeni_siparis_trigger
-- AFTER INSERT ON siparis
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_yeni_siparis_eklendiginde_odeme_ekle();
```

Açıklama: Yeni sipariş eklendiğinde ödeme ekliyor.

```
-- CREATE OR REPLACE FUNCTION trigger_yeni_kullanici_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--     -- Yeni bir kullanıcı eklendiğinde notice yazdır
--     IF TG_OP = 'INSERT' THEN
--         RAISE NOTICE 'Yeni kullanıcı eklendi: % %', NEW.ad, NEW.soyad;
--     END IF;

--     RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;
```

```
-- CREATE TRIGGER yeni_kullanici_ekle_trigger
-- AFTER INSERT ON kullanıcı
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_yeni_kullanici_ekle();
```

Açıklama: Yeni kullanıcı ekliyor.

```
-- CREATE OR REPLACE FUNCTION get_user_age_by_id(
--     IN kullanıcı_id INT
-- ) RETURNS INTEGER AS $$
-- DECLARE
--     user_birth_date DATE;
--     current_date DATE := CURRENT_DATE;
--     user_age INTEGER;
-- BEGIN
--     SELECT dogumTarihi INTO user_birth_date FROM kullanıcı WHERE kullanıcıID =
-- kullanıcı_id;

--     user_age := EXTRACT(YEAR FROM age(current_date, user_birth_date));

--     RETURN user_age;
-- END;
```

```
-- $$ LANGUAGE plpgsql;
```

Açıklama: kullanıcının doğum tarihinden yaşını hesaplıyor.

```
-- CREATE OR REPLACE PROCEDURE delete_expired_coupons() AS $$
-- BEGIN
--   DELETE FROM kupon WHERE sonKullanmaTarihi < CURRENT_DATE;
-- END;
-- $$ LANGUAGE plpgsql;
```

```
-- CREATE OR REPLACE FUNCTION son_oturumu_getir(
--   IN kullanıcı_id INT,
--   OUT bitis_tarihi DATE
-- ) AS $$
-- BEGIN
--   SELECT DATE(bitisZamani)
--   INTO bitis_tarihi
--   FROM kullanıcıOturum
--   WHERE kullanıcıID = kullanıcı_id
--   ORDER BY bitisZamani DESC
--   LIMIT 1;
-- END;
-- $$ LANGUAGE plpgsql;
```

Açıklama: Son oturumun saatini getiriyor.

```
-- CREATE OR REPLACE PROCEDURE urun_fiyat_arttir(IN yuzdelikartis DECIMAL) AS $$
-- BEGIN
--   UPDATE urun SET fiyat = fiyat + (fiyat * yuzdelikartis / 100);
-- END;
-- $$ LANGUAGE plpgsql;
```

```
select*from urun;
call urun_fiyat_arttir(50);
select*from urun;
```

```
select*from kupon;
```

Açıklama: Zam giriyor.

SQL İfadeleri:

```

-- CREATE TABLE kullanici
-- (
--     kullaniciID serial PRIMARY KEY NOT NULL,
--     ad VARCHAR(50) NOT NULL,
--     soyad VARCHAR(50) NOT NULL,
--     eposta VARCHAR(50) NOT NULL,
--     sifre VARCHAR(50) NOT NULL,
--     telefonNo VARCHAR(15),
--     dogumTarihi DATE
-- );

-- CREATE TABLE urun
-- (
--     urunID serial PRIMARY KEY NOT NULL,
--     ad VARCHAR(50) NOT NULL,
--     fiyat money NOT NULL,
--     uretimTarihi DATE NOT NULL
-- );

-- CREATE TABLE siparis
-- (
--     siparisID serial PRIMARY KEY NOT NULL,
--     fiyat money NOT NULL,
--     tarih DATE NOT NULL
-- );

-- CREATE TABLE kargoBilgileri
-- (
--     kargoID serial PRIMARY KEY NOT NULL,
--     takipNo integer NOT NULL,
--     teslimatTarihi date NOT NULL,
--     firmaID INT,
--     FOREIGN KEY (firmaID) REFERENCES kargoFirmasi(firmaID)
-- );

-- CREATE TABLE iadeTalep
-- (
--     talepID serial PRIMARY KEY NOT NULL,
--     Durum boolean,
--     talepTarihi date NOT NULL
-- );

-- CREATE TABLE kategori
-- (

```

```

-- kategoriID serial PRIMARY KEY NOT NULL
-- );

-- CREATE TABLE kullaniciOturum
-- (
--     oturumID serial PRIMARY KEY NOT NULL,
--         baslangicZamani TIMESTAMP,
--         bitisZamani TIMESTAMP
-- );

-- CREATE TABLE fatura
-- (
--     faturalID serial PRIMARY KEY NOT NULL,
--         faturaTarihi date NOT NULL
-- );

-- CREATE TABLE odeme
-- (
--     odemeID serial PRIMARY KEY NOT NULL,
--         fiyat money,
--         odemeTarihi date
-- );

-- CREATE TABLE kartlaOdeme (
--     odemeID serial PRIMARY KEY,
--     taksitMiktari money,
--     CONSTRAINT kartlaOdeme_fk FOREIGN KEY (odemeID) REFERENCES odeme(odemeID)
-- );

-- CREATE TABLE kapidaOdeme (
--     odemeID serial PRIMARY KEY,
--     ekstraKargoUcreti money,
--     CONSTRAINT kapidaOdeme_fk FOREIGN KEY (odemeID) REFERENCES odeme(odemeID)
-- );

-- CREATE TABLE satici
-- (
--     saticiID serial PRIMARY KEY NOT NULL,
--         saticiAdi varchar(50) not null,
--         saticiAdres varchar(100) NOT NULL,
--         telefonNo varchar(11) NOT NULL,
--         eposta varchar(50) NOT NULL
-- );

```

```
-- CREATE TABLE kupon
-- (
--     kuponID serial PRIMARY KEY NOT NULL,
--     sonKullanmaTarihi date not null,
--     indirimMiktari integer NOT NULL
-- );

-- CREATE TABLE kargoFirmasi
-- (
--     firmaID serial PRIMARY KEY NOT NULL,
--     firmaAdi varchar(50) not null,
--     telefonNo varchar(11) NOT NULL
-- );

-- CREATE TABLE saticiUrun
-- (
--     saticiID INT,
--     urunID INT,
--     PRIMARY KEY (saticiID, urunID),
--     FOREIGN KEY (saticiID) REFERENCES satici(saticiID),
--     FOREIGN KEY (urunID) REFERENCES urun(urunID)
-- );

-- CREATE TABLE siparisUrun
-- (
--     siparisID INT,
--     urunID INT,
--     PRIMARY KEY (siparisID, urunID),
--     FOREIGN KEY (urunID) REFERENCES urun(urunID),
--     FOREIGN KEY (siparisID) REFERENCES siparis(siparisID)
-- );

-- CREATE TABLE geriBildirim
-- (
--     geriBildirimID serial PRIMARY KEY NOT NULL,
--     yorum varchar(255) not null,
--     puan SMALLINT NOT NULL
-- )

-- ALTER TABLE geriBildirim
-- ADD COLUMN kullaniciID INT;

-- ALTER TABLE geriBildirim
-- ADD CONSTRAINT FK_geriBildirim_kullanici
```

```
-- FOREIGN KEY (kullaniciID) REFERENCES kullanıcı(kullaniciID);

-- ALTER TABLE kullanıcıOturum
-- ADD COLUMN kullanıcıID INT;

-- ALTER TABLE kullanıcıOturum
-- ADD CONSTRAINT FK_kullanıcıOturum_kullanıcı
-- FOREIGN KEY (kullaniciID) REFERENCES kullanıcı(kullaniciID);

-- ALTER TABLE ödeme
-- ADD COLUMN siparişID INT;

-- ALTER TABLE ödeme
-- ADD CONSTRAINT FK_ödem_sipariş
-- FOREIGN KEY (siparişID) REFERENCES sipariş(siparişID);

-- ALTER TABLE iadeTalep
-- ADD COLUMN kullanıcıID INT;

-- ALTER TABLE iadeTalep
-- ADD CONSTRAINT FK_iadeTalep_kullanıcı
-- FOREIGN KEY (kullaniciID) REFERENCES kullanıcı(kullaniciID);

-- ALTER TABLE sipariş
-- ADD COLUMN kullanıcıID INT;

-- ALTER TABLE sipariş
-- ADD CONSTRAINT FK_sipariş_kullanıcı
-- FOREIGN KEY (kullaniciID) REFERENCES kullanıcı(kullaniciID);

-- ALTER TABLE iadeTalep
-- ADD COLUMN siparişID INT;

-- ALTER TABLE iadeTalep
-- ADD CONSTRAINT FK_iadeTalep_sipariş
-- FOREIGN KEY (siparişID) REFERENCES sipariş(siparişID);

-- ALTER TABLE kupon
-- ADD COLUMN siparişID INT;

-- ALTER TABLE kupon
-- ADD CONSTRAINT FK_kupon_sipariş
-- FOREIGN KEY (siparişID) REFERENCES sipariş(siparişID);
```

```
-- ALTER TABLE fatura
-- ADD COLUMN siparisID INT;

-- ALTER TABLE fatura
-- ADD CONSTRAINT FK_fatura_siparis
-- FOREIGN KEY (siparisID) REFERENCES siparis(siparisID);

-- ALTER TABLE kategori
-- ADD COLUMN urunID INT;

-- ALTER TABLE kategori
-- ADD CONSTRAINT FK_kategori_urun
-- FOREIGN KEY (urunID) REFERENCES urun(urunID);

-- CREATE OR REPLACE FUNCTION trigger_yeni_siparis_eklendiginde_odeme_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--     -- Yeni bir sipariş eklenirse, yeni bir ödeme satırı ekleyin
--     INSERT INTO odeme (fiyat, odemeTarihi, kullanıcıID)
--     VALUES (NEW.fiyat, CURRENT_DATE, NEW.kullanıcıID);

--     RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE OR REPLACE FUNCTION trigger_siparis_ekle_guncelle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--     -- Yeni bir sipariş eklendiğinde veya güncellendiğinde yapılacak işlemler buraya eklenir
--     IF TG_OP = 'INSERT' THEN
--         RAISE NOTICE 'Yeni sipariş eklendi: %', NEW.siparisID;
--     ELSIF TG_OP = 'UPDATE' THEN
--         RAISE NOTICE 'Sipariş güncellendi: %', NEW.siparisID;
--     END IF;

--     RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE TRIGGER siparis_trigger
-- AFTER INSERT OR UPDATE ON siparis
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_siparis_ekle_guncelle();
```

```
-- CREATE OR REPLACE FUNCTION trigger_yeni_kullanici_eklendiginde_oturum_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--     -- Yeni bir kullanıcı eklenirse, yeni bir oturum satırı ekleyin
--     INSERT INTO kullanıcıOturum (başlangıçZamani, bitişZamani, kullanıcıID)
--     VALUES (CURRENT_TIMESTAMP, NULL, NEW.kullanıcıID);

--     RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE TRIGGER yeni_kullanici_trigger
-- AFTER INSERT ON kullanıcı
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_yeni_kullanici_eklendiginde_oturum_ekle();

-- CREATE OR REPLACE FUNCTION trigger_yeni_siparis_eklendiginde_odeme_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--     -- Yeni bir sipariş eklenirse, yeni bir ödeme satırı ekleyin
--     INSERT INTO ödeme (fiyat, ödemeTarihi, siparişID)
--     VALUES (NEW.fiyat, CURRENT_DATE, NEW.siparişID);

--     RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE TRIGGER yeni_siparis_trigger
-- AFTER INSERT ON siparis
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_yeni_siparis_eklendiginde_odeme_ekle();

-- INSERT INTO urun (ad, fiyat, üretimTarihi)
-- VALUES
--     ('Ürün1', 49.99, '2023-01-01'),
--     ('Ürün2', 29.99, '2023-02-01');

-- CREATE OR REPLACE FUNCTION trigger_yeni_kullanici_ekle()
-- RETURNS TRIGGER AS $$
-- BEGIN
--     -- Yeni bir kullanıcı eklendiğinde notice yazdır
--     IF TG_OP = 'INSERT' THEN
--         RAISE NOTICE 'Yeni kullanıcı eklendi: % %', NEW.ad, NEW.soyad;
--     END IF;
```



```

-- RETURN NEW;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE TRIGGER yeni_kullanici_ekle_trigger
-- AFTER INSERT ON kullanici
-- FOR EACH ROW
-- EXECUTE FUNCTION trigger_yeni_kullanici_ekle();

-- CREATE OR REPLACE FUNCTION get_user_age_by_id(
-- IN kullanici_id INT
-- ) RETURNS INTEGER AS $$
-- DECLARE
-- user_birth_date DATE;
-- current_date DATE := CURRENT_DATE;
-- user_age INTEGER;
-- BEGIN
-- SELECT dogumTarihi INTO user_birth_date FROM kullanici WHERE kullaniciID =
kullanici_id;

-- user_age := EXTRACT(YEAR FROM age(current_date, user_birth_date));

-- RETURN user_age;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE OR REPLACE PROCEDURE delete_expired_coupons() AS $$
-- BEGIN
-- DELETE FROM kupon WHERE sonKullanmaTarihi < CURRENT_DATE;
-- END;
-- $$ LANGUAGE plpgsql;

-- CREATE OR REPLACE FUNCTION son_oturumu_getir(
-- IN kullanici_id INT,
-- OUT bitis_tarihi DATE
-- ) AS $$
-- BEGIN
-- SELECT DATE(bitisZamani)
-- INTO bitis_tarihi
-- FROM kullaniciOturum

```

```

-- WHERE kullanıcıID = kullanıcı_id
-- ORDER BY bitisZamani DESC
-- LIMIT 1;
-- END;
-- $$ LANGUAGE plpgsql;

-- SELECT * FROM son_oturumu_getir(14);

-- CREATE OR REPLACE PROCEDURE urun_fiyat_arttir(IN yuzdelikartis DECIMAL) AS $$
-- BEGIN
--     UPDATE urun SET fiyat = fiyat + (fiyat * yuzdelikartis / 100);
-- END;
-- $$ LANGUAGE plpgsql;

select*from urun;
call urun_fiyat_arttir(50);
select*from urun;

select*from kupon;

-- INSERT INTO kupon (sonKullanmaTarihi, indirimMiktari) VALUES
-- ('2023-12-31', 10),
-- ('2023-12-31', 15),
-- ('2023-12-31', 20),
-- ('2023-12-31', 25),
-- ('2023-12-31', 30);

-- INSERT INTO kupon (sonKullanmaTarihi, indirimMiktari) VALUES
-- ('2022-01-01', 5),
-- ('2022-01-01', 12),
-- ('2022-01-01', 18),
-- ('2022-01-01', 22),
-- ('2022-01-01', 28);

-- call delete_expired_coupons();
-- select * from kupon;

-- INSERT INTO siparis (fiyat, tarih)
-- VALUES
-- (59.99, '2023-03-01'), -- Fiyat, tarih ve kullanıcıID'yi uygun şekilde güncelleyin
-- (79.99, '2023-03-02'); -- Fiyat, tarih ve kullanıcıID'yi uygun şekilde güncelleyin

```

```
-- INSERT INTO kullanıcı (ad, soyad, eposta, sifre, telefonNo, dogumTarihi)
-- VALUES
-- ('Ahmet', 'Yılmaz', 'ahmet@email.com', 'ahmet123', '1234567890', '1990-01-15'),
-- ('Ayşe', 'Demir', 'ayse@email.com', 'ayse456', '9876543210', '1985-05-20');
```

Kaynak Kodlar:

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
namespace VYSProje
{
    public partial class Form1 : Form
    {
        string connString = "server=localhost; port=5432; Database=postgres; user ID=postgres;
password=yasincan691";

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }

        private void btnClickKullaniciEkle(object sender, EventArgs e)
        {
            using (var kullaniciEkle = new KullaniciEkle())
            {
                if (kullaniciEkle.ShowDialog() == DialogResult.OK)
                {
                    try
```

```

{
    using (NpgsqlConnection connection = new NpgsqlConnection(connString))
    {
        connection.Open();

        using (NpgsqlCommand cmd = new NpgsqlCommand())
        {
            cmd.Connection = connection;

            string ad = kullanıcıEkle.Ad;
            string soyad = kullanıcıEkle.Soyad;
            string eposta = kullanıcıEkle.Eposta;
            string sifre = kullanıcıEkle.Sifre;
            string telefonNo = kullanıcıEkle.TelefonNo;
            DateTime? dogumTarihi = kullanıcıEkle.DogumTarihi;

            cmd.CommandText = "INSERT INTO kullanıcı(ad, soyad, eposta, sifre,
telefonNo, dogumTarihi) " +
                "VALUES (:ad, :soyad, :eposta, :sifre, :telefonNo, :dogumTarihi)";

            cmd.Parameters.AddWithValue(":ad", ad);
            cmd.Parameters.AddWithValue(":soyad", soyad);
            cmd.Parameters.AddWithValue(":eposta", eposta);
            cmd.Parameters.AddWithValue(":sifre", sifre);
            cmd.Parameters.AddWithValue(":telefonNo", telefonNo);
            cmd.Parameters.AddWithValue(":dogumTarihi", dogumTarihi);

            int rowsAffected = cmd.ExecuteNonQuery();

            if (rowsAffected > 0)
            {
                MessageBox.Show("Kullanıcı başarıyla eklendi.");
            }
            else
            {
                MessageBox.Show("Kullanıcı eklenirken bir hata oluştu.");
            }
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message);
}

```

```
}  
}  
}  
}
```

```
private void btnClickSiparisEkle(object sender, EventArgs e)  
{  
    using (var siparisEkle = new SiparisEkle())  
    {  
        if (siparisEkle.ShowDialog() == DialogResult.OK)  
        {  
            try  
            {  
                using (NpgsqlConnection connection = new NpgsqlConnection(connString))  
                {  
                    connection.Open();  
  
                    using (NpgsqlCommand cmd = new NpgsqlCommand())  
                    {  
                        cmd.Connection = connection;  
  
                        decimal fiyat = siparisEkle.Fiyat;  
                        DateTime tarih = siparisEkle.Tarih;  
  
                        cmd.CommandText = "INSERT INTO siparis(fiyat, tarih) " +  
                            "VALUES (:fiyat, :tarih)";  
  
                        cmd.Parameters.AddWithValue(":fiyat", fiyat);  
                        cmd.Parameters.AddWithValue(":tarih", tarih);  
  
                        int rowsAffected = cmd.ExecuteNonQuery();  
  
                        if (rowsAffected > 0)  
                        {  
                            MessageBox.Show("Sipariş başarıyla eklendi.");  
                        }  
                        else  
                        {  
                            MessageBox.Show("Sipariş eklenirken bir hata oluştu.");  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

```

    }
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message);
}
}
}
}
}

```

```

private void btnClickKullaniciSil(object sender, EventArgs e)
{
    using (var kullaniciSil = new KullaniciSil())
    {
        if (kullaniciSil.ShowDialog() == DialogResult.OK)
        {
            try
            {
                int kullaniciIDToDelete = kullaniciSil.KullaniciID; // Get the user ID to delete

                using (NpgsqlConnection connection = new NpgsqlConnection(connString))
                {
                    connection.Open();

                    using (NpgsqlCommand cmd = new NpgsqlCommand())
                    {
                        cmd.Connection = connection;

                        cmd.CommandText = "DELETE FROM kullanici WHERE kullaniciID =
:kullaniciID";

                        cmd.Parameters.AddWithValue(":kullaniciID", kullaniciIDToDelete);

                        int rowsAffected = cmd.ExecuteNonQuery();

                        if (rowsAffected > 0)
                        {
                            MessageBox.Show("Kullanıcı başarıyla silindi.");
                        }
                        else
                        {
                            MessageBox.Show("Kullanıcı silinirken bir hata oluştu veya belirtilen
kullanıcı bulunamadı.");
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Hata: " + ex.Message);
    }
    }
    }
}

```

```

private void btnClickKullaniciGuncelle(object sender, EventArgs e)
{
    using (var kullaniciGuncelle = new KullaniciGuncelle())
    {
        if (kullaniciGuncelle.ShowDialog() == DialogResult.OK)
        {
            try
            {
                int kullaniciIDToUpdate = kullaniciGuncelle.KullaniciID;

                using (NpgsqlConnection connection = new NpgsqlConnection(connString))
                {
                    connection.Open();

                    using (NpgsqlCommand cmd = new NpgsqlCommand())
                    {
                        cmd.Connection = connection;

                        string ad = kullaniciGuncelle.Ad;
                        string soyad = kullaniciGuncelle.Soyad;
                        string eposta = kullaniciGuncelle.Eposta;
                        string sifre = kullaniciGuncelle.Sifre;
                        string telefonNo = kullaniciGuncelle.TelefonNo;
                        DateTime? dogumTarihi = kullaniciGuncelle.DogumTarihi;

                        cmd.CommandText = "UPDATE kullanici " +
                            "SET ad = :ad, soyad = :soyad, eposta = :eposta, sifre = :sifre,
telefonNo = :telefonNo, dogumTarihi = :dogumTarihi " +
                            "WHERE kullaniciID = :kullaniciID";

                        cmd.Parameters.AddWithValue(":ad", ad);

```

```

cmd.Parameters.AddWithValue(":soyad", soyad);
cmd.Parameters.AddWithValue(":eposta", eposta);
cmd.Parameters.AddWithValue(":sifre", sifre);
cmd.Parameters.AddWithValue(":telefonNo", telefonNo);
cmd.Parameters.AddWithValue(":dogumTarihi", dogumTarihi);
cmd.Parameters.AddWithValue(":kullaniciID", kullaniciIDToUpdate);

int rowsAffected = cmd.ExecuteNonQuery();

if (rowsAffected > 0)
{
    MessageBox.Show("Kullanıcı başarıyla güncellendi.");
}
else
{
    MessageBox.Show("Kullanıcı güncellenirken bir hata oluştu veya belirtilen
kullanıcı bulunamadı.");
}
}
}
}
}
catch (Exception ex)
{
    MessageBox.Show("Hata: " + ex.Message);
}
}
}
}
private void btnGetUserAge_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtUserId.Text, out int userId))
    {
        try
        {
            using (NpgsqlConnection connection = new NpgsqlConnection(connString))
            {
                connection.Open();

                using (NpgsqlCommand cmd = new NpgsqlCommand())
                {
                    cmd.Connection = connection;
                    cmd.CommandText = "SELECT get_user_age_by_id(@userId)";
                    cmd.Parameters.AddWithValue("@userId", userId);

```



```

        int userAge = Convert.ToInt32(cmd.ExecuteScalar());

        txtResult.Text = $"{userId} li kullanıcı {userAge} yaşında.";
    }
}
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message);
}
}
else
{
    MessageBox.Show("Please enter a valid numeric user ID.");
}
}

private void btnClickKullaniciAra(object sender, EventArgs e)
{
    using (var kullaniciAra = new KullaniciAra())
    {
        if (kullaniciAra.ShowDialog() == DialogResult.OK)
        {
            try
            {
                int kullaniciIDAra = kullaniciAra.KullaniciID;

                using (NpgsqlConnection connection = new NpgsqlConnection(connString))
                {
                    connection.Open();

                    using (NpgsqlCommand cmd = new NpgsqlCommand())
                    {
                        cmd.Connection = connection;

                        cmd.CommandText = "SELECT kullaniciID, ad, soyad, eposta, telefonNo,
dogumTarihi " +
                            "FROM kullanici " +
                            "WHERE kullaniciID = :kullaniciID";

                        cmd.Parameters.AddWithValue(":kullaniciID", kullaniciIDAra);

```

```

using (NpgsqlDataReader reader = cmd.ExecuteReader())
{
    if (reader.Read())
    {
        int kullanicilD = reader.GetInt32(0);
        string ad = reader.GetString(1);
        string soyad = reader.GetString(2);
        string eposta = reader.GetString(3);
        string telefonNo = reader.GetString(4);

        MessageBox.Show($"Kullanıcı Bulundu:\n" +
            $"ID: {kullanicilD}\n" +
            $"Ad: {ad}\n" +
            $"Soyad: {soyad}\n" +
            $"Eposta: {eposta}\n" +
            $"Telefon: {telefonNo}\n");
    }
    else
    {
        MessageBox.Show("Belirtilen kullanıcı bulunamadı.");
    }
}
}
}
}
}
}
}
}
}
}

private void btnGetLastSession_Click(object sender, EventArgs e)
{
    if (int.TryParse(txtUserIdOturum.Text, out int userId))
    {
        try
        {
            using (NpgsqlConnection connection = new NpgsqlConnection(connString))

```

```

{
    connection.Open();

    using (NpgsqlCommand cmd = new NpgsqlCommand())
    {
        cmd.Connection = connection;
        cmd.CommandText = "SELECT * FROM son_oturumu_getir(@userId)";
        cmd.Parameters.AddWithValue("@userId", userId);

        using (NpgsqlDataReader reader = cmd.ExecuteReader())
        {
            if (reader.Read())
            {
                DateTime bitisZamani = reader.GetDateTime(0);

                txtResultOturum.Text = $"{userId} ID ye sahip kullanıcının oturumunun
bitis zamanı:\n" +
                    $" {bitisZamani}\n";
            }
            else
            {
                txtResultOturum.Text = $"No session found for user with ID {userId}.";
            }
        }
    }
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message);
}
else
{
    MessageBox.Show("Please enter a valid numeric user ID.");
}
}

```

```

private void btnDeleteExpiredCoupons_Click(object sender, EventArgs e)
{
    try
    {
        using (NpgsqlConnection connection = new NpgsqlConnection(connString))
        {

```

```

connection.Open();

using (NpgsqlCommand cmd = new NpgsqlCommand())
{
    cmd.Connection = connection;
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "CALL delete_expired_coupons()";

    cmd.ExecuteNonQuery();

    MessageBox.Show("Expired coupons deleted successfully.");
}
}
}
catch (Exception ex)
{
    MessageBox.Show("Error: " + ex.Message);
}
}

```

```

private void btnIncreaseProductPrices_Click(object sender, EventArgs e)
{
    try
    {
        decimal yuzdelikArtis = 10;

        using (NpgsqlConnection connection = new NpgsqlConnection(connString))
        {
            connection.Open();

            using (NpgsqlCommand cmd = new NpgsqlCommand())
            {
                cmd.Connection = connection;
                cmd.CommandType = CommandType.Text;
                cmd.CommandText = $"CALL urun_fiyat_arttir({yuzdelikArtis})";

                cmd.ExecuteNonQuery();

                MessageBox.Show($"Product prices increased by {yuzdelikArtis}%
successfully.");
            }
        }
    }
}

```

```

        catch (Exception ex)
        {
            MessageBox.Show("Error: " + ex.Message);
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace VYSProje
{
    public partial class KullaniciAra : Form
    {
        public int KullaniciID { get; private set; }

        public KullaniciAra()
        {
            InitializeComponent();
        }

        private void btnAra_Click(object sender, EventArgs e)
        {
            if (int.TryParse(txtKullaniciID.Text, out int kullaniciID))
            {
                KullaniciID = kullaniciID;
                DialogResult = DialogResult.OK;
                Close();
            }
            else
            {
                MessageBox.Show("Geçerli bir kullanıcı ID girin.");
            }
        }
    }
}

```

```

    }

    private void KullaniciAra_Load(object sender, EventArgs e)
    {

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace VYSProje
{
    public partial class KullaniciEkle : Form
    {
        public string Ad { get; private set; }
        public string Soyad { get; private set; }
        public string Eposta { get; private set; }
        public string Sifre { get; private set; }
        public string TelefonNo { get; private set; }
        public DateTime? DogumTarihi { get; private set; }
        public KullaniciEkle()
        {
            InitializeComponent();
        }

        private void KullaniciEkleOnayla(object sender, EventArgs e)
        {
            Ad = txtAd.Text;
            Soyad = txtSoyad.Text;
            Eposta = txtEposta.Text;
            Sifre = txtSifre.Text;
            TelefonNo = txtTelNo.Text;

```

```

        DogumTarihi = dateTimePicker1.Value;

        DialogResult = DialogResult.OK;
        Close();
    }

    private void KullaniciEkle_Load(object sender, EventArgs e)
    {

    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace VYSProje
{
    public partial class KullaniciGuncelle : Form
    {
        public int KullaniciID { get; private set; }
        public string Ad { get; private set; }
        public string Soyad { get; private set; }
        public string Eposta { get; private set; }
        public string Sifre { get; private set; }
        public string TelefonNo { get; private set; }
        public DateTime? DogumTarihi { get; private set; }

        public KullaniciGuncelle()
        {
            InitializeComponent();
        }

        private void btnGuncelle_Click(object sender, EventArgs e)
        {
            if (int.TryParse(txtKullaniciID.Text, out int kullaniciID))
            {

```

```

        KullaniciID = kullaniciID;
        Ad = txtAd.Text;
        Soyad = txtSoyad.Text;
        Eposta = txtEposta.Text;
        Sifre = txtSifre.Text;
        TelefonNo = txtTelefonNo.Text;

        if (DateTime.TryParse(dtpDogumTarihi.Text, out DateTime dogumTarihi))
        {
            DogumTarihi = dogumTarihi;
        }
        else
        {
            DogumTarihi = null;
        }

        DialogResult = DialogResult.OK;
        Close();
    }
    else
    {
        MessageBox.Show("Geçerli bir kullanıcı ID girin.");
    }
}

private void KullaniciGuncelle_Load(object sender, EventArgs e)
{
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace VYSProje

```



```

{
    public partial class KullaniciSil : Form
    {
        public int KullaniciID { get; private set; }

        public KullaniciSil()
        {
            InitializeComponent();
        }

        private void btnSil_Click(object sender, EventArgs e)
        {
            if (int.TryParse(txtKullaniciID.Text, out int kullaniciID))
            {
                KullaniciID = kullaniciID;
                DialogResult = DialogResult.OK;
                Close();
            }
            else
            {
                MessageBox.Show("Geçerli bir kullanıcı ID girin.");
            }
        }

        private void KullaniciSil_Load(object sender, EventArgs e)
        {
        }
    }
}

```

```

using System;
using System.Windows.Forms;

```

```

namespace VYSProje
{
    public partial class SiparisEkle : Form
    {
        public decimal Fiyat { get; private set; }
        public DateTime Tarih { get; private set; }

        public SiparisEkle()
        {

```

```
        InitializeComponent();
    }

    private void SiparisEkleOnayla(object sender, EventArgs e)
    {
        Fiyat = decimal.Parse(txtFiyat.Text);
        Tarih = dateTimePicker1.Value;

        DialogResult = DialogResult.OK;
        Close();
    }

    private void SiparisEkle_Load(object sender, EventArgs e)
    {
    }
}
}
```