Perde ve Ev Tekstili Satış Yönetim Sistemi

Kullanıcı Rolleri ve Yetkileri:

1. Müşteriler

- **Ürünleri Görüntüleme:** Mevcut ürünleri ve stok durumunu inceleyebilir.
- **Sipariş Verme:** Sadece stokta mevcut ürünler için sipariş oluşturabilir.
- **Sipariş Takibi:** Kendi siparişlerinin durumunu görüntüleyebilir.
- **Yetki Sınırlamaları:** Bilgi güncelleme yetkisi yoktur, yalnızca okuma ve sipariş oluşturma yetkisine sahiptir.

2. Çalışanlar (Sipariş ve Stok Yöneticileri)

- **Stok Yönetimi:** Ürün stok seviyelerini görüntüleme ve güncelleme yetkisine sahiptir.
- **Sipariş Durumu Yönetimi:** Siparişlerin durumunu güncelleyebilir.
- **Tedarik Süreci Yönetimi:** Ürün tedarikçilerini yönetebilir ve stok seviyelerini düzenleyebilir.
- **Yetki Kapsamı:** Stok, sipariş bilgilerine tam erişim; müşteri bilgilerine sınırlı erişim.

3. Yöneticiler

- **Tam Yetki:** Tüm veri türlerini (stok, sipariş, müşteri, tedarikçi) yönetebilir.
- **Yeni Eklemeler:** Ürün, tedarikçi ve kategori ekleme yetkisine sahiptir.
- **Raporlama:** Stok durumu, tedarik performansı ve sipariş durumları hakkında rapor oluşturabilir.

VARLIKLAR

Müşteriler

- **MüşteriID (PK): Benzersiz müşteri kimliği.**
- **Ad: Müşterinin adı.**
- **Soyad: Müşterinin soyadı.**
- **İletişim Bilgileri: Telefon, e-posta gibi müşteri iletişim bilgileri.**

Siparişler

- **SiparişID (PK): Benzersiz sipariş kimliği.**
- **MüşteriID (FK): Siparişi veren müşteri kimliği.**
- **Sipariş Tarihi: Siparişin verildiği tarih.**
- **Durum: Siparişin durumu (Örneğin: Beklemede, Tamamlandı, İptal Edildi).**

Sipariş Detayları

- **SiparişDetayID (PK): Benzersiz sipariş detayı kimliği.**
- **SiparişID (FK): Siparişin kimliği.**
- **ÜrünID (FK): Siparişte yer alan ürünün kimliği.**
- **Miktar: Sipariş edilen ürün miktarı.**

Ürünler

- **ÜrünID (PK): Benzersiz ürün kimliği.**
- **Ad: Ürün adı.**
- **Fiyat: Ürün fiyatı.**
- **Stok: Ürünün mevcut stok miktarı.**
- **TedarikçiID (FK): Ürünün tedarikçisi.**
- **KategoriID (FK): Ürünün kategorisi.**

Tedarikçiler

- **TedarikçiID (PK): Benzersiz tedarikçi kimliği.**
- **Ad: Tedarikçi adı.**
- **İletişim Bilgileri: Tedarikçi iletişim bilgileri (örneğin telefon, e-posta).**

Kategoriler

- **KategoriID (PK): Benzersiz kategori kimliği.**
- **Ad: Kategori adı (örneğin Perde, Nevresim).**

Ödemeler

- **ÖdemeID (PK): Benzersiz ödeme kimliği.**

- **SiparişID (FK): Ödeme yapılan sipariş kimliği.**
- **Ödeme Türü: Ödeme türü (örneğin kredi kartı, nakit).**
- **Ödeme Tarihi: Ödeme yapıldığı tarih.**

Çalışanlar

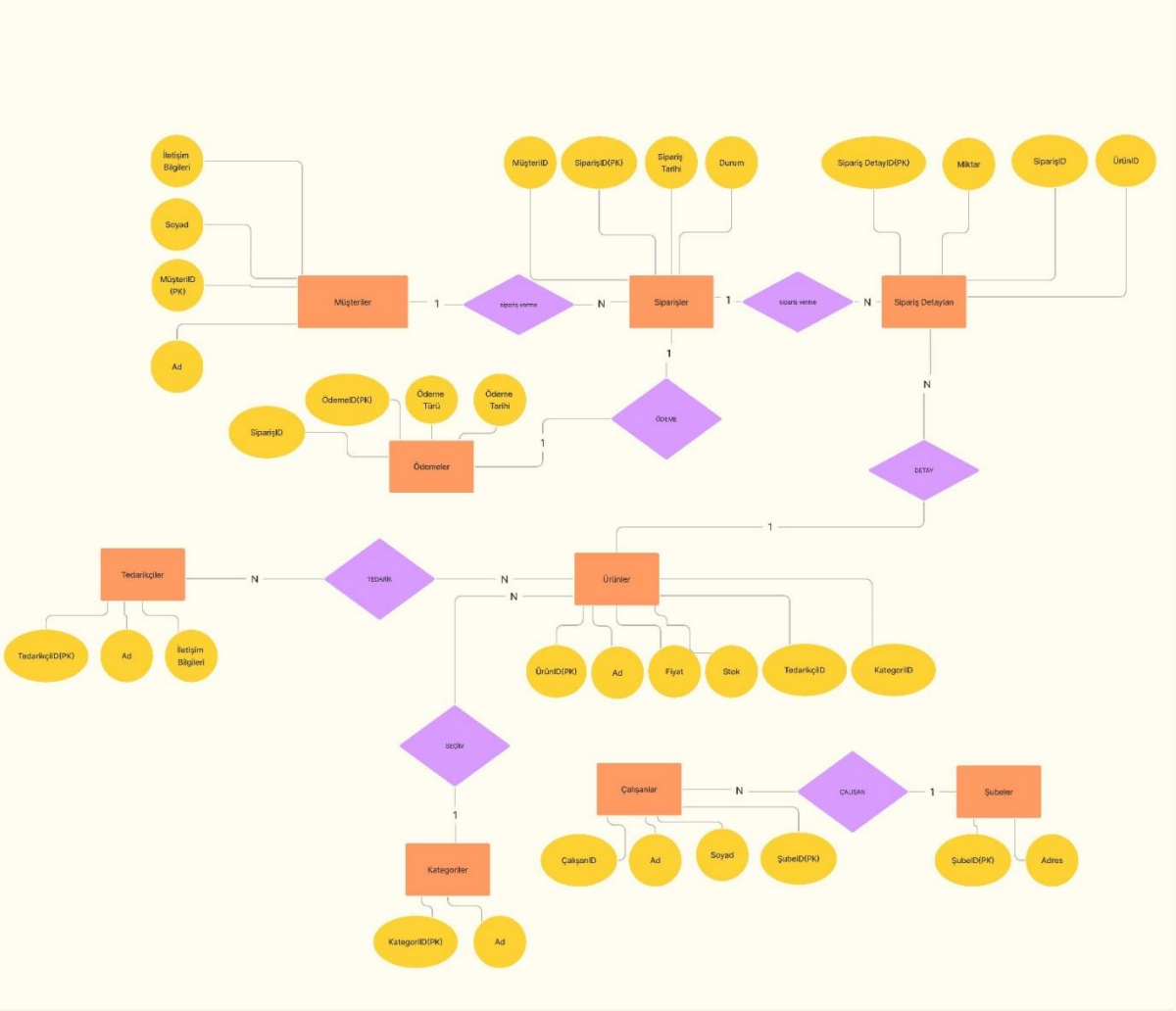
- **ÇalışanID (PK): Benzersiz çalışan kimliği.**
- **Ad: Çalışan adı.**
- **Soyad: Çalışan soyadı.**
- **ŞubeID (FK): Çalışanın görevli olduğu şube.**

Şubeler

- **ŞubeID (PK): Benzersiz şube kimliği.**
- **Adres: Şubenin adresi.**

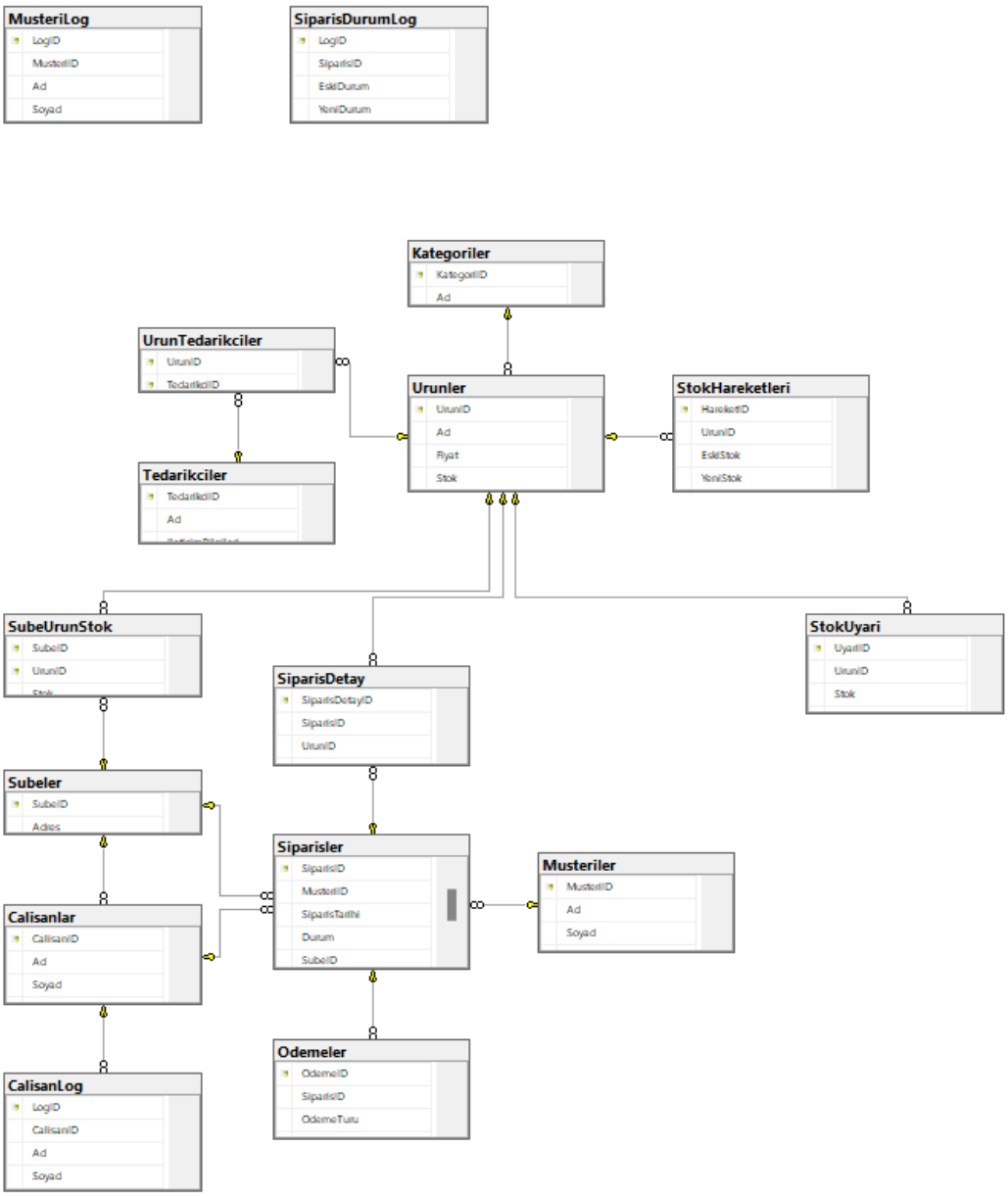
İlişkiler ve Sayısal Kısıtlamalar

- **Müşteriler → Siparişler: Her müşteri birden fazla sipariş verebilir ancak her sipariş yalnızca bir müşteriye ait olabilir. Bu ilişki 1:N'dir.**
- **Siparişler → Sipariş Detayları: Her sipariş birden fazla sipariş detayına sahip olabilir. Her sipariş detayı ise yalnızca bir siparişe bağlıdır. Bu ilişki 1:N'dir.**
- **Ürünler → Sipariş Detayları: Her sipariş detayı bir ürüne bağlıdır, ancak bir ürün birden fazla sipariş detayı içerebilir. Bu ilişki 1:N'dir.**
- **Ürünler → Tedarikçiler: Her ürün yalnızca bir tedarikçi tarafından sağlanır, ancak bir tedarikçi birden fazla ürün sağlayabilir. Bu ilişki N:N'dir.**
- **Ürünler → Kategoriler: Her ürün yalnızca bir kategoriye ait olabilir, ancak bir kategori birden fazla ürüne sahip olabilir. Bu ilişki 1:N'dir.**
- **Siparişler → Ödemeler: Her sipariş yalnızca bir ödeme kaydı içerebilir, ancak bir ödeme yalnızca bir siparişe bağlı olabilir. Bu ilişki 1:1'dir.**



İlişki Kümesi:

Müşteriler(MusteriID,Ad,Soyad,iletişimBilgileri)



Tabloların Oluşturulması

```
-- Müşteriler tablosu oluşturma

CREATE TABLE Musteriler (
    MusteriID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(50) NOT NULL,
    Soyad NVARCHAR(50) NOT NULL,
    IletisimBilgileri NVARCHAR(255)
```

```
);

-- Şubeler tablosu oluşturma
CREATE TABLE Subeler (
    SubeID INT PRIMARY KEY IDENTITY(1,1),
    Adres NVARCHAR(255) NOT NULL
);


-- Çalışanlar tablosu oluşturma
CREATE TABLE Calisanlar (
    CalisanID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(50) NOT NULL,
    Soyad NVARCHAR(50) NOT NULL,
    SubeID INT NOT NULL,       -- Çalışan ile şube arasındaki ilişki
    FOREIGN KEY (SubeID) REFERENCES Subeler(SubeID)
);



-- Siparişler tablosu oluşturma
CREATE TABLE Siparisler (
    SiparisID INT PRIMARY KEY IDENTITY(1,1),
    MusteriID INT NOT NULL,
    SiparisTarihi DATE NOT NULL,
    Durum NVARCHAR(50),
    SubeID INT NOT NULL,       -- Şube ile ilişki
    CalisanID INT,             -- Çalışan ile ilişki
    FOREIGN KEY (MusteriID) REFERENCES Musteriler(MusteriID),
    FOREIGN KEY (SubeID) REFERENCES Subeler(SubeID),
    FOREIGN KEY (CalisanID) REFERENCES Calisanlar(CalisanID)
);

-- Ödemeler tablosu oluşturma
CREATE TABLE Odemeler (
    OdemeID INT PRIMARY KEY IDENTITY(1,1),
    SiparisID INT NOT NULL,
    OdemeTuru NVARCHAR(50),
    OdemeTarihi DATE,
    FOREIGN KEY (SiparisID) REFERENCES Siparisler(SiparisID)
);
```

```sql
-- Kategoriler tablosu oluşturma
CREATE TABLE Kategoriler (
    KategoriID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(50) NOT NULL
);

-- Ürünler tablosu oluşturma
CREATE TABLE Urunler (
    UrunID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(50) NOT NULL,
    Fiyat DECIMAL(10, 2) NOT NULL,
    Stok INT NOT NULL,
    KategoriID INT NOT NULL,
    FOREIGN KEY (KategoriID) REFERENCES Kategoriler(KategoriID)
);

-- Tedarikçiler tablosu oluşturma
CREATE TABLE Tedarikciler (
    TedarikciID INT PRIMARY KEY IDENTITY(1,1),
    Ad NVARCHAR(50) NOT NULL,
    IletisimBilgileri NVARCHAR(255)
);

--  Ürünler ve Tedarikçiler için ara tablo oluşturma
CREATE TABLE UrunTedarikciler (
    UrunID INT NOT NULL,
    TedarikciID INT NOT NULL,
    PRIMARY KEY (UrunID, TedarikciID),
    FOREIGN KEY (UrunID) REFERENCES Urunler(UrunID),
    FOREIGN KEY (TedarikciID) REFERENCES Tedarikciler(TedarikciID)
);

-- Sipariş Detayları tablosu oluşturma
CREATE TABLE SiparisDetay (
    SiparisDetayID INT PRIMARY KEY IDENTITY(1,1),
    SiparisID INT NOT NULL,
    UrunID INT NOT NULL,
    Miktar INT NOT NULL,
    FOREIGN KEY (SiparisID) REFERENCES Siparisler(SiparisID),
    FOREIGN KEY (UrunID) REFERENCES Urunler(UrunID)
);
```

```sql
-- Şube-Ürün Stok Takibi için ara tablo oluşturma
CREATE TABLE SubeUrunStok (
    SubeID INT NOT NULL,
    UrunID INT NOT NULL,
    Stok INT NOT NULL,
    PRIMARY KEY (SubeID, UrunID),
    FOREIGN KEY (SubeID) REFERENCES Subeler(SubeID),
    FOREIGN KEY (UrunID) REFERENCES Urunler(UrunID)
);
-- MusteriLog Tablosunu Oluşturma

CREATE TABLE MusteriLog (
    LogID INT PRIMARY KEY IDENTITY(1,1),
    MusteriID INT,
    Ad NVARCHAR(50),
    Soyad NVARCHAR(50),
    IslemTarihi DATETIME
);

--SiparisDurumLog Tablosunu Oluşturma
CREATE TABLE SiparisDurumLog (
    LogID INT PRIMARY KEY IDENTITY(1,1),
    SiparisID INT NOT NULL,
    EskiDurum NVARCHAR(50),
    YeniDurum NVARCHAR(50),
    GuncellenmeTarihi DATETIME
);



CREATE TABLE StokHareketleri (
    HareketID INT PRIMARY KEY IDENTITY(1,1),
    UrunID INT NOT NULL,
    EskiStok INT,
    YeniStok INT,
    HareketTarihi DATETIME,
    FOREIGN KEY (UrunID) REFERENCES Urunler(UrunID)
);
CREATE TABLE StokUyari (
    UyariID INT PRIMARY KEY IDENTITY(1,1),
    UrunID INT NOT NULL,
    Stok INT,
    UyariTarihi DATETIME,
```

```
    FOREIGN KEY (UrunID) REFERENCES Urunler(UrunID)
);

CREATE TABLE CalisanLog (
    LogID INT PRIMARY KEY IDENTITY(1,1),
    CalisanID INT,
    Ad NVARCHAR(50),
    Soyad NVARCHAR(50),
    GuncellenmeTarihi DATETIME,
    FOREIGN KEY (CalisanID) REFERENCES Calisanlar(CalisanID)
);
```

NORMALİZASYON SÜRECİ

1NF (Birinci Normal Form)

Kural:

- Her bir hücre atomik olmalıdır (bölünemez).
- Tekrarlayan grup veya çok değerli alan olmamalıdır.

Tablolar için 1NF Analizi:

1. **Müşteriler**:
    - Her müşteri bir ID, ad ve soyad içermektedir. Hücreler atomiktir. 1NF'e uygun.
2. **Siparişler**:
    - Her sipariş bir ID'ye, müşteriye, çalışan bilgisine ve sipariş tarihine sahiptir. Hücreler atomiktir, 1NF'e uygun.
3. **Ürünler**:
    - Her ürün için ID, ad, fiyat ve stok atomik. 1NF'e uygun.
4. **Tedarikçiler**:
    - Her tedarikçi tek bir ID ve ad ile temsil ediliyor. Atomik, 1NF'e uygun.
5. **SiparişDetay**:
    - Her sipariş detayında ürün, sipariş ve miktar belirtilmiş. Atomik, 1NF'e uygun.

2NF (İkinci Normal Form)

Kural:

- 1NF'e uygun olmalıdır.
- Tüm tabloların birincil anahtarı (primary key) tüm belirleyici olmayan sütunları belirleyebilmelidir.
- Kısmi bağımlılıklar kaldırılmalıdır.

Tablolar için 2NF Analizi:

1. **Müşteriler**:
    - Birincil anahtar (MüşteriID) tüm sütunları belirler. Kısmi bağımlılık yok. 2NF'e uygun.
2. **Siparişler**:
    - Birincil anahtar (SiparişID), müşteriyi, çalışanı ve tarihi belirliyor. Kısmi bağımlılık yok. 2NF'e uygun.
3. **Ürünler**:
    - Birincil anahtar (UrunID), adı, fiyatı ve stoku belirliyor. Kısmi bağımlılık yok. 2NF'e uygun.
4. **Tedarikçiler**:
    - Birincil anahtar (TedarikciID), adı belirler. Kısmi bağımlılık yok. 2NF'e uygun.
5. **SiparişDetay**:
    - Birincil anahtar (SiparişDetayID), sipariş, ürün ve miktar bilgilerini belirler. Kısmi bağımlılık yok. 2NF'e uygun.
6. **SubeUrunStok**:
    - Birincil anahtar (ŞubeID + UrunID) birleşik anahtardır. Stok miktarı, bu birleşik anahtara tamamen bağlıdır. Kısmi bağımlılık yok. 2NF'e uygun.

3NF (Üçüncü Normal Form)

Kural:

- 2NF'e uygun olmalıdır.

- Transitif bağımlılıklar kaldırılmalıdır (Bir sütun başka bir sütuna bağımlı olmamalıdır).

Tablolar için 3NF Analizi:

1. **Müşteriler**:
    - Müşteri bilgileri arasında transitif bağımlılık yok. 3NF'e uygun.
2. **Siparişler**:
    - Sipariş bilgileri arasında transitif bağımlılık yok. 3NF'e uygun.
3. **Ürünler**:
    - Ürün fiyatı, stok ve adı arasında transitif bağımlılık yok. 3NF'e uygun.
4. **Tedarikçiler**:
    - Tedarikçi bilgileri arasında transitif bağımlılık yok. 3NF'e uygun.
5. **SiparişDetay**:
    - Sipariş ve ürün bilgileri ayrı tablolar aracılığıyla tanımlanıyor. Transitif bağımlılık yok. 3NF'e uygun.
6. **SubeUrunStok**:
    - Şube ve ürün stok bilgileri arasında transitif bağımlılık yok. 3NF'e uygun.

BCNF (Boyce-Codd Normal Form)

Kural:

- 3NF'e uygun olmalıdır.
- Her belirleyici (determinant) bir aday anahtar olmalıdır.

Tablolar için BCNF Analizi:

1. **Müşteriler**:
    - MüşteriID, tek aday anahtar olarak tüm alanları belirler. BCNF'e uygun.
2. **Siparişler**:
    - SiparişID, tek aday anahtar olarak tüm alanları belirler. BCNF'e uygun.
3. **Ürünler**:
    - UrunID, tek aday anahtar olarak tüm alanları belirler. BCNF'e uygun.

4. **Tedarikçiler**:
    - TedarikciID, tek aday anahtar olarak tüm alanları belirler. BCNF'e uygun.
5. **SiparişDetay**:
    - SiparişDetayID, tek aday anahtar olarak tüm alanları belirler. BCNF'e uygun.
6. **SubeUrunStok**:
    - ŞubeID ve UrunID birleşik anahtar, stok miktarını belirler. BCNF'e uygun.

```
-- TABLOLARA VERİ EKLEME

--  Musteriler tablosu veri ekleme
INSERT INTO Musteriler (Ad, Soyad, IletisimBilgileri) VALUES
('Ahmet', 'Yilmaz', 'ahmet.yilmaz@example.com'),
('Ayse', 'Demir', 'ayse.demir@example.com'),
('Mehmet', 'Kaya', 'mehmet.kaya@example.com'),
('Fatma', 'Celik', 'fatma.celik@example.com'),
('Ali', 'Can', 'ali.can@example.com'),
('Burcu', 'Dogan', 'burcu.dogan@example.com'),
('Cem', 'Karaca', 'cem.karaca@example.com'),
('Derya', 'Aydin', 'derya.aydin@example.com'),
('Eren', 'Yildirim', 'eren.yildirim@example.com'),
('Selin', 'Orhan', 'selin.orhan@example.com'),
('Tuncay', 'Arslan', 'tuncay.arslan@example.com'),
('Sibel', 'Ekinci', 'sibel.ekinci@example.com'),
('Okan', 'Kurt', 'okan.kurt@example.com'),
('Nazan', 'Guler', 'nazan.guler@example.com'),
('Yusuf', 'Ozkan', 'yusuf.ozkan@example.com'),
('Aylin', 'Bozkurt', 'aylin.bozkurt@example.com'),
('Serkan', 'Aksoy', 'serkan.aksoy@example.com'),
('Betul', 'Yildiz', 'betul.yildiz@example.com'),
('Oguz', 'Turan', 'oguz.turan@example.com');

--  Subeler tablosu veri ekleme
INSERT INTO Subeler (Adres) VALUES
('Istanbul - Kadikoy'),
('Ankara - Cankaya'),
('Izmir - Konak'),
('Bursa - Nilüfer'),
('Antalya - Muratpasa'),
```

```
('Eskisehir - Odunpazari'),
('Trabzon - Ortahisar'),
('Adana - Seyhan'),
('Kayseri - Melikgazi'),
('Gaziantep - Sahinbey');

--  Calisanlar tablosu veri ekleme
INSERT INTO Calisanlar (Ad, Soyad, SubeID) VALUES
('Ebru', 'Altun', 1),
('Murat', 'Gul', 1),
('Zeynep', 'Korkmaz', 2),
('Hakan', 'Sahin', 2),
('Selin', 'Tekin', 3),
('Can', 'Ersoy', 4),
('Cansu', 'Acar', 4),
('Merve', 'Aslan', 5),
('Umut', 'Cetin', 5),
('Gizem', 'Kilic', 3),
('Deniz', 'Celik', 6),
('Kerem', 'Yilmaz', 7),
('Melis', 'Duman', 8),
('Furkan', 'Bas', 9),
('Irem', 'Ozkan', 10);

--  Siparisler tablosu veri ekleme
INSERT INTO Siparisler (MusteriID, SiparisTarihi, Durum, SubeID, CalisanID) VALUES
(1, '2025-01-01', 'Hazirlaniyor', 1, 1),
(2, '2025-01-02', 'Tamamlandi', 2, 3),
(3, '2025-01-03', 'Teslim Edildi', 3, 5),
(4, '2025-01-04', 'Hazirlaniyor', 4, 6),
(5, '2025-01-05', 'Tamamlandi', 5, 8),
(6, '2025-01-06', 'Teslim Edildi', 1, 2),
(7, '2025-01-07', 'Hazirlaniyor', 2, 4),
(8, '2025-01-08', 'Tamamlandi', 3, 10),
(9, '2025-01-09', 'Teslim Edildi', 4, 7),
(10, '2025-01-10', 'Hazirlaniyor', 5, 9),
(11, '2025-01-11', 'Tamamlandi', 6, 11),
(12, '2025-01-12', 'Teslim Edildi', 7, 12);

--  Odemeler tablosu veri ekleme
INSERT INTO Odemeler (SiparisID, OdemeTuru, OdemeTarihi) VALUES
(1, 'Kredi Karti', '2025-01-01'),
(2, 'Nakit', '2025-01-02'),
```

```
(3, 'Havale', '2025-01-03'),
(4, 'Kredi Karti', '2025-01-04'),
(5, 'Nakit', '2025-01-05'),
(6, 'Havale', '2025-01-06'),
(7, 'Kredi Karti', '2025-01-07'),
(8, 'Nakit', '2025-01-08'),
(9, 'Havale', '2025-01-09'),
(10, 'Kredi Karti', '2025-01-10'),
(11, 'Nakit', '2025-01-11'),
(12, 'Havale', '2025-01-12');

-- Kategoriler tablosu veri ekleme
INSERT INTO Kategoriler (Ad) VALUES
('Perde'),
('Nevresim Takımı'),
('Yatak Örtüsü'),
('Havlu'),
('Banyo Halısı'),
('Yastık'),
('Stor Perde'),
('Tül Perde'),
('Fon Perde');

--  Urunler tablosu veri ekleme
INSERT INTO Urunler (Ad, Fiyat, Stok, KategoriID) VALUES
('Dantel Perde', 500, 100, 1),
('Saten Nevresim', 750, 50, 2),
('Pamuklu Yatak Örtüsü', 900, 40, 3),
('El Havlusu', 50, 300, 4),
('Kaymaz Taban Halı', 450, 20, 5),
('Ortopedik Yastık', 200, 80, 6),
('Keten Stor Perde', 600, 30, 7),
('Modern Tül Perde', 350, 50, 8),
('Desenli Fon Perde', 800, 25, 9);

--  Tedarikciler tablosu veri ekleme
INSERT INTO Tedarikciler (Ad, IletisimBilgileri) VALUES
('Perdeci Tedarik', 'perdeci.tedarik@example.com'),
('EvTek Tedarik', 'evtek.tedarik@example.com'),
('Tekstil Lojistik', 'tekstil.lojistik@example.com'),
('Mega Ev', 'mega.ev@example.com'),
('Dekor Tedarik', 'dekor.tedarik@example.com'),
('Evim Tedarik', 'evim.tedarik@example.com'),
```

```
('Moda Ev Tekstil', 'moda.evtekstil@example.com');

--  UrunTedarikciler tablosu veri ekleme
INSERT INTO UrunTedarikciler (UrunID, TedarikciID) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 1),
(8, 2),
(9, 3);

--  SiparisDetay tablosu veri ekleme
INSERT INTO SiparisDetay (SiparisID, UrunID, Miktar) VALUES
(1, 1, 2),
(1, 2, 1),
(2, 3, 2),
(3, 4, 3),
(4, 5, 1),
(5, 6, 2),
(6, 7, 1),
(7, 8, 3),
(8, 9, 2),
(9, 1, 1),
(10, 2, 2),
(11, 3, 4),
(12, 4, 1);

--  SubeUrunStok tablosu veri ekleme
INSERT INTO SubeUrunStok (SubeID, UrunID, Stok) VALUES
(1, 1, 20),
(1, 2, 30),
(2, 3, 25),
(3, 4, 50),
(4, 5, 15),
(5, 6, 10),
(6, 7, 12),
(7, 8, 20),
(8, 9, 5),
(9, 1, 10),
(10, 2, 15);
```

```sql
-- SORGULAMA KODLARI

-- Belirli bir müşteriyi ismine göre arama
SELECT * FROM Musteriler WHERE Ad = 'Ahmet';

-- Belirli bir müşteri tarafından verilen siparişleri listeleme
SELECT * FROM Siparisler WHERE MusteriID = 1;

-- Belirli bir fiyattan daha pahalı ürünleri listeleme
SELECT * FROM Urunler WHERE Fiyat > 500;

-- Her bir şubede stokta bulunan ürün miktarını listeleme
SELECT SubeID, UrunID, Stok FROM SubeUrunStok;

-- Her bir şubede çalışan çalışanları listeleme
SELECT * FROM Calisanlar WHERE SubeID = 1;

-- Sipariş detayları ile birlikte tüm siparişleri listeleme
SELECT s.SiparisID, s.MusteriID, s.SiparisTarihi, sd.UrunID, sd.Miktar
FROM Siparisler s
JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID;


-- Her bir müşterinin yaptığı toplam harcamayı hesaplama
SELECT m.MusteriID, m.Ad, m.Soyad, SUM(u.Fiyat * sd.Miktar) AS ToplamHarcamalar
FROM Musteriler m
JOIN Siparisler s ON m.MusteriID = s.MusteriID
JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID
JOIN Urunler u ON sd.UrunID = u.UrunID
GROUP BY m.MusteriID, m.Ad, m.Soyad;

-- En çok satan ürünler
SELECT TOP 5 u.UrunID, u.Ad, SUM(sd.Miktar) AS ToplamSatis
FROM Urunler u
JOIN SiparisDetay sd ON u.UrunID = sd.UrunID
GROUP BY u.UrunID, u.Ad
ORDER BY ToplamSatis DESC;
```

```sql
-- Belirli bir tarih aralığındaki toplam satışlar
SELECT SUM(u.Fiyat * sd.Miktar) AS ToplamSatis
FROM Siparisler s
JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID
JOIN Urunler u ON sd.UrunID = u.UrunID
WHERE s.SiparisTarihi BETWEEN '2025-01-01' AND '2025-01-31';

-- Her bir çalışan tarafından gerçekleştirilen satışlar
SELECT c.CalisanID, c.Ad, c.Soyad, SUM(u.Fiyat * sd.Miktar) AS ToplamSatis
FROM Calisanlar c
JOIN Siparisler s ON c.CalisanID = s.CalisanID
JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID
JOIN Urunler u ON sd.UrunID = u.UrunID
GROUP BY c.CalisanID, c.Ad, c.Soyad;

-- En çok alışveriş yapan müşteriler
SELECT TOP 5 m.MusteriID, m.Ad, m.Soyad, SUM(u.Fiyat * sd.Miktar) AS ToplamHarcamalar
FROM Musteriler m
JOIN Siparisler s ON m.MusteriID = s.MusteriID
JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID
JOIN Urunler u ON sd.UrunID = u.UrunID
GROUP BY m.MusteriID, m.Ad, m.Soyad
ORDER BY ToplamHarcamalar DESC;
```

- ———
- ———

```sql
-- Yeni Müşteri Ekleme
CREATE PROCEDURE spYeniMusteriEkle
    @Ad NVARCHAR(50),
    @Soyad NVARCHAR(50),
    @IletisimBilgileri NVARCHAR(255)
AS
BEGIN
    INSERT INTO Musteriler (Ad, Soyad, IletisimBilgileri)
    VALUES (@Ad, @Soyad, @IletisimBilgileri);
END
 -- saklı yordamı çalıştırma
```

```
 EXEC spYeniMusteriEkle @Ad = 'Deniz', @Soyad = 'Demir', @IletisimBilgileri =
'deniz.demir@example.com';


--Sipariş Ekleme
CREATE PROCEDURE spSiparisEkle
    @MusteriID INT,
    @SiparisTarihi DATE,
    @Durum NVARCHAR(50),
    @SubeID INT,
    @CalisanID INT
AS
BEGIN
    INSERT INTO Siparisler (MusteriID, SiparisTarihi, Durum, SubeID, CalisanID)
    VALUES (@MusteriID, @SiparisTarihi, @Durum, @SubeID, @CalisanID);
END
-- saklı yordamı çalıştırma
EXEC spSiparisEkle @MusteriID = 1, @SiparisTarihi = '2025-01-15', @Durum =
'Hazirlaniyor', @SubeID = 1, @CalisanID = 1;

--Ürün Satışlarını Getirme
CREATE PROCEDURE spUrunSatislari
    @UrunID INT
AS
BEGIN
    SELECT sd.SiparisID, sd.Miktar, sd.UrunID, s.SiparisTarihi
    FROM SiparisDetay sd
    JOIN Siparisler s ON sd.SiparisID = s.SiparisID
    WHERE sd.UrunID = @UrunID;
END
 -- saklı yordamı çalıştırma
 EXEC spUrunSatislari @UrunID = 1;


-- Müşteri Detaylı Bilgilerini Getiren Stored Procedure

CREATE PROCEDURE spMusteriDetay
    @MusteriID INT
AS
BEGIN
    -- Müşteri bilgilerini al
    SELECT MusteriID, Ad, Soyad, IletisimBilgileri
    FROM Musteriler
    WHERE MusteriID = @MusteriID;
```

```sql
    -- Müşterinin siparişlerini al
    SELECT s.SiparisID, s.SiparisTarihi, s.Durum, s.SubeID, s.CalisanID
    FROM Siparisler s
    WHERE s.MusteriID = @MusteriID;

    -- Müşterinin toplam harcamalarını hesapla
    SELECT SUM(u.Fiyat * sd.Miktar) AS ToplamHarcamalar
    FROM Siparisler s
    JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID
    JOIN Urunler u ON sd.UrunID = u.UrunID
    WHERE s.MusteriID = @MusteriID;
END
-- saklı yordamı çalıştırma
EXEC spMusteriDetay @MusteriID = 1;

-- Şube Performans Raporu Oluşturan Stored Procedure
CREATE PROCEDURE spSubePerformansRaporu
AS
BEGIN
    SELECT s.SubeID, SUM(u.Fiyat * sd.Miktar) AS ToplamSatisTutari, COUNT(DISTINCT
s.SiparisID) AS ToplamSiparisSayisi
    FROM Siparisler s
    JOIN SiparisDetay sd ON s.SiparisID = sd.SiparisID
    JOIN Urunler u ON sd.UrunID = u.UrunID
    GROUP BY s.SubeID
    ORDER BY ToplamSatisTutari DESC;
END
-- saklı yordamı çalıştırma
EXEC spSubePerformansRaporu;

-- Ürün Kategorilerine Göre Satış Raporu Oluşturan Stored Procedure
CREATE PROCEDURE spKategoriBazliSatisRaporu
AS
BEGIN
    SELECT k.Ad AS KategoriAdi, SUM(u.Fiyat * sd.Miktar) AS ToplamSatisTutari,
COUNT(DISTINCT sd.SiparisID) AS ToplamSiparisSayisi
    FROM Urunler u
    JOIN Kategoriler k ON u.KategoriID = k.KategoriID
    JOIN SiparisDetay sd ON u.UrunID = sd.UrunID
    GROUP BY k.Ad
    ORDER BY ToplamSatisTutari DESC;
```

```
END
-- saklı yordamı çalıştırma
EXEC spKategoriBazliSatisRaporu;
```

--TRİGGERS

- **Tanım:** Belirli bir tablo üzerinde bir olay (INSERT, UPDATE, DELETE) gerçekleştiğinde otomatik olarak çalışan bir SQL komutudur.
- **Kullanım:** Veri bütünlüğünü sağlamak veya işlemleri izlemek için kullanılır.

```
-- Stok Güncelleme Trigger
CREATE TRIGGER trgStokGuncelle
ON SiparisDetay
AFTER INSERT
AS
BEGIN
    UPDATE Urunler
    SET Stok = Stok - inserted.Miktar
    FROM Urunler u
    JOIN inserted ON u.UrunID = inserted.UrunID;
END;

-- Sipariş Durumu Güncelleme Trigger
CREATE TRIGGER trgSiparisTamamlandi
ON Odemeler
AFTER INSERT
AS
BEGIN
    UPDATE Siparisler
    SET Durum = 'Tamamlandi'
    WHERE SiparisID IN (SELECT SiparisID FROM inserted);
END;

-- Müşteri Log Oluşturma Trigger
CREATE TRIGGER trgMusteriLog
ON Musteriler
AFTER INSERT
AS
BEGIN
    INSERT INTO MusteriLog (MusteriID, Ad, Soyad, IslemTarihi)
```

```
    SELECT MusteriID, Ad, Soyad, GETDATE()
    FROM inserted;
END;

-- Kritik Stok Seviyesi Uyarı Trigger
CREATE TRIGGER trgStokKritikSeviye
ON Urunler
AFTER UPDATE
AS
BEGIN
    IF EXISTS (SELECT 1 FROM inserted WHERE Stok < 5)
    BEGIN
        INSERT INTO StokUyari (UrunID, Stok, UyariTarihi)
        SELECT UrunID, Stok, GETDATE()
        FROM inserted
        WHERE Stok < 5;
    END;
END;

-- Çalışan Güncelleme Log Trigger
CREATE TRIGGER trgCalisanGuncelleLog
ON Calisanlar
AFTER UPDATE
AS
BEGIN
    INSERT INTO CalisanLog (CalisanID, Ad, Soyad, GuncellenmeTarihi)
    SELECT CalisanID, Ad, Soyad, GETDATE()
    FROM inserted;
END;

-- Sipariş Ödeme Trigger
CREATE TRIGGER trgOdemeEkle
ON Siparisler
AFTER INSERT
AS
BEGIN
    INSERT INTO Odemeler (SiparisID, OdemeTuru, OdemeTarihi)
    SELECT SiparisID, 'Bekleniyor', NULL
    FROM inserted;
END;

-- Ürün Silme Trigger
CREATE TRIGGER trgUrunSil
```

```
ON Urunler
AFTER DELETE
AS
BEGIN
    DELETE FROM SiparisDetay WHERE UrunID IN (SELECT UrunID FROM deleted);
    DELETE FROM UrunTedarikciler WHERE UrunID IN (SELECT UrunID FROM deleted);
END;

-- Müşteri Silme Trigger
CREATE TRIGGER trgMusteriSilIlişkiliKayitlarSil
ON Musteriler
AFTER DELETE
AS
BEGIN
    DELETE sd
    FROM SiparisDetay sd
    JOIN Siparisler s ON sd.SiparisID = s.SiparisID
    WHERE s.MusteriID IN (SELECT MusteriID FROM deleted);

    DELETE FROM Odemeler
    WHERE SiparisID IN (SELECT SiparisID FROM Siparisler WHERE MusteriID IN (SELECT
MusteriID FROM deleted));

    DELETE FROM Siparisler WHERE MusteriID IN (SELECT MusteriID FROM deleted);
END;

-- Stok Kontrol Trigger
CREATE TRIGGER trgStokKontrol
ON Urunler
AFTER UPDATE
AS
BEGIN
    INSERT INTO StokHareketleri (UrunID, EskiStok, YeniStok, HareketTarihi)
    SELECT d.UrunID, d.Stok, i.Stok, GETDATE()
    FROM deleted d
    JOIN inserted i ON d.UrunID = i.UrunID;

    IF EXISTS (SELECT 1 FROM inserted WHERE Stok < 10)
    BEGIN
        INSERT INTO StokUyari (UrunID, Stok, UyariTarihi)
        SELECT UrunID, Stok, GETDATE()
        FROM inserted
        WHERE Stok < 10;
```

```
    END;
END;

-- Sipariş Durumu ve Log Trigger
CREATE TRIGGER trgSiparisDurumGuncelleLog
ON Siparisler
AFTER UPDATE
AS
BEGIN
    INSERT INTO SiparisDurumLog (SiparisID, EskiDurum, YeniDurum, GuncellenmeTarihi)
    SELECT d.SiparisID, d.Durum, i.Durum, GETDATE()
    FROM deleted d
    JOIN inserted i ON d.SiparisID = i.SiparisID;
END;

-- Sipariş Stok Güncelleme Trigger
CREATE TRIGGER trgSiparisEkleStokGuncelle
ON SiparisDetay
AFTER INSERT
AS
BEGIN
    UPDATE Urunler
    SET Stok = Stok - i.Miktar
    FROM Urunler u
    JOIN inserted i ON u.UrunID = i.UrunID;
END;
```

--TRANSACTİONLAR

- **Tanım:** Bir grup SQL komutunun tümünün başarıyla çalışması ya da hiçbirinin çalışmaması gereken bir birimdir.
- **Kullanım:** Veri bütünlüğünü ve tutarlılığı sağlamak için kullanılır.

```
-- Sipariş ve Ödeme Eklemek için Transaction
BEGIN TRANSACTION;
```

```sql
BEGIN TRY
    -- Sipariş ekleme
    INSERT INTO Siparisler (MusteriID, SiparisTarihi, Durum, SubeID, CalisanID)
    VALUES (1, '2025-01-15', 'Hazirlaniyor', 1, 1);

    -- Eklenen siparişin ID'sini al
    DECLARE @SiparisID INT;
    SET @SiparisID = SCOPE_IDENTITY();

    -- Ödeme ekleme
    INSERT INTO Odemeler (SiparisID, OdemeTuru, OdemeTarihi)
    VALUES (@SiparisID, 'Kredi Karti', '2025-01-15');

    -- İşlemleri onayla
    COMMIT TRANSACTION;
    PRINT 'Sipariş ve ödeme başarıyla eklendi.';
END TRY
BEGIN CATCH
    -- Hata durumunda işlemleri geri al
    ROLLBACK TRANSACTION;
    PRINT 'Hata oluştu. İşlemler geri alındı.';
    THROW;
END CATCH;



-- Ürün Stoğunu Güncelleme ve Şube Stok Kaydı
BEGIN TRANSACTION;

BEGIN TRY
    -- Ürün stok miktarını azalt
    UPDATE Urunler
    SET Stok = Stok - 5
    WHERE UrunID = 1;

    -- Şube stok miktarını azalt
    UPDATE SubeUrunStok
    SET Stok = Stok - 5
    WHERE SubeID = 1 AND UrunID = 1;

    -- İşlemleri onayla
    COMMIT TRANSACTION;
    PRINT 'Ürün ve şube stokları başarıyla güncellendi.';
END TRY
```

```
BEGIN CATCH
    -- Hata durumunda işlemleri geri al
    ROLLBACK TRANSACTION;
    PRINT 'Hata oluştu. İşlemler geri alındı.';
    THROW;
END CATCH;


--Sipariş Durumu Güncelleme ve Log Kaydı

BEGIN TRANSACTION;

BEGIN TRY
    -- Sipariş durumunu güncelle
    UPDATE Siparisler
    SET Durum = 'Tamamlandi'
    WHERE SiparisID = 1;

    -- Log kaydı ekle
    INSERT INTO SiparisDurumLog (SiparisID, EskiDurum, YeniDurum, GuncellenmeTarihi)
    VALUES (1, 'Hazirlaniyor', 'Tamamlandi', GETDATE());

    -- İşlemleri onayla
    COMMIT TRANSACTION;
    PRINT 'Sipariş durumu başarıyla güncellendi ve log kaydı eklendi.';
END TRY
BEGIN CATCH
    -- Hata durumunda işlemleri geri al
    ROLLBACK TRANSACTION;
    PRINT 'Hata oluştu. İşlemler geri alındı.';
    THROW;
END CATCH;


-- Stok Transferi (Bir Şubeden Diğerine)
BEGIN TRANSACTION;

BEGIN TRY
    -- Şubeden stok çıkar
    UPDATE SubeUrunStok
    SET Stok = Stok - 10
    WHERE SubeID = 1 AND UrunID = 1;

    -- Diğer şubeye stok ekle
    UPDATE SubeUrunStok
```

```
    SET Stok = Stok + 10
    WHERE SubeID = 2 AND UrunID = 1;

    -- İşlemleri onayla
    COMMIT TRANSACTION;
    PRINT 'Stok transferi başarıyla gerçekleştirildi.';
END TRY
BEGIN CATCH
    -- Hata durumunda işlemleri geri al
    ROLLBACK TRANSACTION;
    PRINT 'Hata oluştu. İşlemler geri alındı.';
    THROW;
END CATCH;



--Müşteri ve İlgili Siparişlerin Silinmesi

BEGIN TRANSACTION;

BEGIN TRY
    -- 1. Ödemeleri sil
    DELETE FROM Odemeler
    WHERE SiparisID IN (
        SELECT SiparisID FROM Siparisler WHERE MusteriID = 1
    );

    -- 2. Sipariş detaylarını sil
    DELETE FROM SiparisDetay
    WHERE SiparisID IN (
        SELECT SiparisID FROM Siparisler WHERE MusteriID = 1
    );

    -- 3. Sipariş loglarını sil
    DELETE FROM SiparisDurumLog
    WHERE SiparisID IN (
        SELECT SiparisID FROM Siparisler WHERE MusteriID = 1
    );

    -- 4. Siparişleri sil
    DELETE FROM Siparisler
    WHERE MusteriID = 1;

    -- 5. Müşteriyi sil
```

```
    DELETE FROM Musteriler
    WHERE MusteriID = 1;

    -- İşlemleri onayla
    COMMIT TRANSACTION;
    PRINT 'Müşteri ve ilişkili tüm kayıtlar başarıyla silindi.';
END TRY

BEGIN CATCH
    -- Hata durumunda işlemleri geri al
    ROLLBACK TRANSACTION;
    PRINT 'Hata oluştu. Tüm işlemler geri alındı.';
    THROW; -- Hata detaylarını yeniden fırlat
END CATCH;
```