

## VERİ TABANI PROJESİ RAPORU

Yemek Sipariş Sistemi

Hazırlayanlar:

Emirhan Yentur-225260062

Eyüp Gök-215260042

Emirhan Güney-225260008

Tarih: 05.01.2025

## İçindekiler

1. Proje Tanımı .....	Sayfa 2
2. ER Diyagramı .....	Sayfa 3
3. Tablo Yapıları .....	Sayfa 4
4. Kod Yapıları ve Fonksiyonlar .....	Sayfa 6
5.Projede Kullanılan SQL Komutları.....	Sayfa 6-26

## 1. Proje Tanımı

Bu proje, yemek sipariş sistemi için veri tabanı tasarımını ve uygulamasını ele almaktadır. Sistemin temel gayesi, kullanıcıların restoranlardan yemek siparişi vermelerini ve bu siparişlerin takibini güvenli bir şekilde kolaylaştırmaktır. Proje, kullanıcılar, restoranlar, yemekler, siparişler ve ödeme işlemleri gibi ana bileşenler içermektedir.

## 2. ER Diyagramı

ER diyagramında aşağıdaki temel varlıklar ve ilişkiler tanımlanmıştır:

- 1. Kullanıcılar:** Sisteme kayıt olan kullanıcı bilgilerini tutar.
- 2. Restoranlar:** Sistem üzerinde hizmet veren restoranların bilgilerini muhafaza eder.
- 3. Yemekler:** Restoranlar tarafından sunulan yemeklerin adı.
- 4. Kategoriler:** Yemeklerin kategorilere göre sınıflandırıldığı yapı.
- 5. Siparişler:** Kullanıcıların verdiği sipariş bilgilerini tutar.
- 6. Sipariş Detayları:** Siparişlerde yer alan her bir ürünün detay bilgileri.
- 7. Teslimatlar:** Sipariş teslimat bilgilerini kayıt altına alır.
- 8. Ödemeler:** Siparişlerin ödeme bilgilerini saklar.
- 9. Puanlamalar:** Kullanıcıların restoranlara yaptıkları puanlama ve yorumlar.
- 10. Yemek Kategori:** Restoranlar tarafından sunulan yemeklerin sınıflandırılması.

## 3. Tablo Yapıları

Aşağıdaki tablolar, projenin temel veri tabanı yapısını oluşturmaktadır:

### 1. Kullanıcılar:

- Alanlar: (KullanıcıID(PK) , isim, Eposta, Telefon, Şifre, Adres, SiparişAdet)

### 2. Restoranlar:

- Alanlar: (RestoranID(PK) , RestoranAdi, Adres, Telefon, MutfakTuru, ÇalışmaSaatleri, Sahip)

### 3. Kategoriler:

- Alanlar: ( KategoriID(PK) , KategoriAdi)

### 4. Yemekler:

- Alanlar: ( YemekID(PK) , YemekAdi, Açıklama, Fiyat, RestoranID )

### 5. Siparişler:

- Alanlar: ( SiparişID(PK) , KullanıcıID , SiparişZamanı, TeslimZamanı, Durum, ToplamFiyat, TeslimatAdresi)

### 6. Sipariş Detayları:

- Alanlar: ( SiparişDetayID(PK) , SiparişID , YemekID , YemekFiyatı, Miktar)

### 7. Teslimatlar:

- Alanlar: ( TeslimatID(PK) , SiparişID , TeslimatDurumu, KuryeAdi)

### 8. Ödemeler:

- Alanlar: ( ÖdemeID(PK) , SiparişID , ÖdemeTuru, ÖdemeBilgileri, ÖdemeDurumu, Tarih)

### 9. Puanlamalar:

- Alanlar: ( PuanlamaID(PK) , KullanıcıID , RestoranID , Puan, Yorum, Tarih)

### 10. Yemek\_Kategori:

- Alanlar: ( YemekID(PK) , KategoriID(PK) )

## 4. Kod Yapıları ve Fonksiyonlar

### 1. Tablo Oluşturma

"Tablo\_Oluşturma.sql" dosyası, sistemin temel veri tabanı yapısını kurmak için gerekli SQL komutlarını barındırır. Bu komutlar doğru bir şekilde çalıştırıldığında, veri tabanı altyapısı eksiksiz bir şekilde oluşturulur.

### 2. Örnek Veri Ekleme

"Örnek\_Veri.sql" dosyasında, her tablo için test senaryolarında kullanılabilecek örnek kayıtlar yer alır. Bu kayıtlar, sistemin işleyişinin kontrol edilmesi ve doğrulanması için kullanılabilir.

### 3. Saklı Prosedürler

"Procedure.sql" dosyasında, sistemde sıkça ihtiyaç duyulan işlemleri kolaylaştıran prosedürler tanımlanmıştır. Örnek olarak:

- Yeni bir sipariş ekleme işlemi.
- Belirli bir kullanıcıya ait siparişlerin listelenmesi.

### 4. Tetikleyici Mekanizmaları

"Trigger.sql" dosyası, veri tabanının da gerçekleşen INSERT, UPDATE veya DELETE gibi işlemler sırasında devreye giren tetikleyicileri içerir. Bu mekanizmalar, sistemin dinamik yapısının korunmasına olanak sağlar.

- Örnek: Yeni bir sipariş eklenmesi durumunda, kullanıcının toplam sipariş sayısının otomatik olarak güncellenmesi sağlanır. Bu sayede verilerin güncel ve tutarlı kalması sağlanır.

### 5. Transaction Yönetimi

"Transaction.sql" dosyası, birden fazla işlemten oluşan kritik süreçlerde veri bütünlüğünü sağlamak amacıyla transaction yapılarını içerir. Bu yapı, rollback mekanizması sayesinde, hata durumunda tüm işlemleri geri alarak sistemin güvenilirliğini korur.

- Örnek: Yeni bir sipariş oluşturulduğunda, sipariş bilgileri teslimat sürecine iletilirken, aynı anda kullanıcının toplam sipariş sayısı otomatik olarak güncellenir. Ayrıca, teslimatla ilgili bilgiler sisteme kaydedilir. Ancak, herhangi bir hata meydana gelirse, tüm işlemler geri alınarak sistem eski, tutarlı haline döner. Bu süreç, veri tabanı bütünlüğünü korurken aynı zamanda kullanıcı deneyimini de geliştirir.

## 5.Projede Kullanılan SQL Komutları

### 1-tablo\_Olusturma.sql

```
CREATE TABLE Kullanici (  
  
    KullaniciID INT PRIMARY KEY IDENTITY(1,1),  
  
    Isim VARCHAR(100) NOT NULL,  
  
    Eposta VARCHAR(100) UNIQUE NOT NULL,  
  
    Telefon VARCHAR(15),  
  
    Sifre VARCHAR(100) NOT NULL,  
  
    Adres TEXT,  
  
    SiparisAdet INT DEFAULT 0 NOT NULL  
  
);
```

```
CREATE TABLE Restoran (  
  
    RestoranID INT PRIMARY KEY IDENTITY(1,1),  
  
    RestoranAdi VARCHAR(100) NOT NULL,  
  
    Adres TEXT NOT NULL,  
  
    Telefon VARCHAR(15),  
  
    MutfakTuru VARCHAR(50),  
  
    CalismaSaatleri VARCHAR(100)  
  
);
```

```
CREATE TABLE Kategori (  
  
    KategoriID INT PRIMARY KEY IDENTITY(1,1),  
  
    KategoriAdi VARCHAR(50) NOT NULL  
  
);
```

```
CREATE TABLE Yemek (  
    YemekID INT PRIMARY KEY IDENTITY(1,1),  
    YemekAdi VARCHAR(100) NOT NULL,  
    Aciklama TEXT,  
    Fiyat DECIMAL(10, 2) NOT NULL,  
    RestoranID INT NOT NULL,  
    FOREIGN KEY (RestoranID) REFERENCES Restoran(RestoranID)  
);
```

```
CREATE TABLE Siparis (  
    SiparisID INT PRIMARY KEY IDENTITY(1,1),  
    KullaniciID INT NOT NULL,  
    SiparisZamani DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    TeslimZamani DATETIME,  
    Durum VARCHAR(50) NOT NULL CHECK (Durum IN ('Hazırlanıyor', 'Yolda', 'Teslim  
Edildi', 'Başarısız')) DEFAULT 'Hazırlanıyor',  
    ToplamFiyat DECIMAL(10, 2) NOT NULL,  
    TeslimatAdresi TEXT NOT NULL,  
    FOREIGN KEY (KullaniciID) REFERENCES Kullanici(KullaniciID)  
);
```

```
CREATE TABLE SiparisDetay (  
    SiparisDetayID INT PRIMARY KEY IDENTITY(1,1),  
    SiparisID INT NOT NULL,  
    YemekID INT NOT NULL,  
    YemekFiyati DECIMAL(10, 2) NOT NULL,
```

```
Miktar INT NOT NULL,  
  
FOREIGN KEY (SiparisID) REFERENCES Siparis(SiparisID),  
  
FOREIGN KEY (YemekID) REFERENCES Yemek(YemekID)  
);
```

```
CREATE TABLE Teslimat (  
  
    TeslimatID INT PRIMARY KEY IDENTITY(1,1),  
  
    SiparisID INT NOT NULL,  
  
    TeslimatDurumu VARCHAR(50) NOT NULL CHECK (TeslimatDurumu IN ('Hazırlanıyor',  
'Yolda', 'Teslim Edildi', 'Başarısız')) DEFAULT 'Hazırlanıyor',  
  
    KuryeAdi VARCHAR(100),  
  
    FOREIGN KEY (SiparisID) REFERENCES Siparis(SiparisID)  
);
```

```
CREATE TABLE Odeme (  
  
    OdemeID INT PRIMARY KEY IDENTITY(1,1),  
  
    SiparisID INT NOT NULL,  
  
    OdemeTuru VARCHAR(50) NOT NULL CHECK (OdemeTuru IN ('Nakit', 'Kredi Kartı',  
'Online')) DEFAULT 'Nakit',  
  
    OdemeBilgileri TEXT,  
  
    OdemeDurumu VARCHAR(50) NOT NULL CHECK (OdemeDurumu IN ('Bekliyor',  
'Tamamlandı')) DEFAULT 'Bekliyor',  
  
    Tarih DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  
    FOREIGN KEY (SiparisID) REFERENCES Siparis(SiparisID)  
);
```

```
CREATE TABLE Puanlama (  

```



```
PuanlamaID INT PRIMARY KEY IDENTITY(1,1),
KullaniciID INT NOT NULL,
RestoranID INT NOT NULL,
Puan INT CHECK (Puan BETWEEN 1 AND 5),
Yorum TEXT,
Tarih DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
FOREIGN KEY (KullaniciID) REFERENCES Kullanici(KullaniciID),
FOREIGN KEY (RestoranID) REFERENCES Restoran(RestoranID)
);
```

```
CREATE TABLE Yemek_Kategori (
    YemekID INT NOT NULL,
    KategoriID INT NOT NULL,
    PRIMARY KEY (YemekID, KategoriID),
    FOREIGN KEY (YemekID) REFERENCES Yemek(YemekID),
    FOREIGN KEY (KategoriID) REFERENCES Kategori(KategoriID)
);
```

## 2-trigger.sql

----sipariş oluşturulduğunda teslimat tablosunda otomatik olarak tablo oluşturulur

```
CREATE TRIGGER trg_InsertTeslimat
ON Siparis
AFTER INSERT
AS
BEGIN
```

```
INSERT INTO Teslimat (SiparisID, TeslimatDurumu, KuryeAdi)
SELECT i.SiparisID, 'Hazırlanıyor', NULL
FROM inserted i;
END;
```

```
INSERT INTO Siparis (KullaniciID, SiparisZamani, TeslimatAdresi, ToplamFiyat)
VALUES (1, GETDATE(), 'Test Adresi, Test Şehir', 100.50);
```

```
SELECT * FROM Siparis
```

```
SELECT * FROM Teslimat
```

---- teslimat tamamlandığında sipariş tablosundaki teslim tarihi düzenlenir

```
CREATE TRIGGER trg_TeslimatTamamlandi
```

```
ON Teslimat
```

```
AFTER UPDATE
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    IF EXISTS (SELECT 1 FROM inserted WHERE TeslimatDurumu = 'Teslim Edildi')
```

```
    BEGIN
```

```
        UPDATE Siparis
```

```
        SET
```

```
Durum = 'Teslim Edildi',  
TeslimZamani = GETDATE()  
FROM Siparis S  
INNER JOIN inserted I ON S.SiparisID = I.SiparisID  
WHERE I.TeslimatDurumu = 'Teslim Edildi';  
END  
END;
```

```
UPDATE Teslimat  
SET TeslimatDurumu = 'Teslim Edildi'  
WHERE TeslimatID = 2;
```

---- sipariş adet için trigger

```
CREATE TRIGGER SiparisAdetArttir  
ON Siparis  
AFTER INSERT  
AS  
BEGIN  
    UPDATE Kullanici  
    SET SiparisAdet = SiparisAdet + 1  
    FROM Kullanici  
    INNER JOIN Inserted i ON Kullanici.KullaniciID = i.KullaniciID;  
END;
```

```
SELECT * FROM Kullanici
```

```
INSERT INTO Siparis (KullaniciID, SiparisZamani, TeslimatAdresi, ToplamFiyat)  
VALUES (1, GETDATE(), 'Deneme Adresi', 200.00);
```

---- teslimat durumu değiştiğinde durumun değişmesi

```
CREATE TRIGGER Trigger_TeslimatDurumuDegistigindeSiparisDurumunuGuncelle  
ON Teslimat  
AFTER UPDATE  
AS  
BEGIN  
    IF UPDATE(TeslimatDurumu)  
    BEGIN  
        DECLARE @SiparisID INT;  
        DECLARE @YeniDurum VARCHAR(50);  
  
        SELECT @SiparisID = SiparisID, @YeniDurum = TeslimatDurumu  
        FROM inserted;  
  
        UPDATE Siparis  
        SET Durum = @YeniDurum  
        WHERE SiparisID = @SiparisID;  
    END  
END;
```

---- teslimat başarısız olduğunda sipariş adet 1 düşer

```
CREATE TRIGGER trg_TeslimatDurumu
```

```
ON Teslimat
```

```
AFTER UPDATE
```

```
AS
```

```
BEGIN
```

```
    IF EXISTS (SELECT 1 FROM inserted WHERE TeslimatDurumu = 'Teslim Edildi')
```

```
    BEGIN
```

```
        UPDATE Siparis
```

```
        SET Durum = 'Teslim Edildi'
```

```
        FROM Siparis s
```

```
        INNER JOIN inserted i ON s.SiparisID = i.SiparisID;
```

```
    END
```

```
IF EXISTS (SELECT 1 FROM inserted WHERE TeslimatDurumu = 'Başarısız')
```

```
BEGIN
```

```
    UPDATE Siparis
```

```
    SET Durum = 'Başarısız'
```

```
    FROM Siparis s
```

```
    INNER JOIN inserted i ON s.SiparisID = i.SiparisID;
```

```
UPDATE Kullanici
```

```
SET SiparisAdet = SiparisAdet - 1
```

```
FROM Kullanici k
INNER JOIN Siparis s ON s.KullaniciID = k.KullaniciID
INNER JOIN inserted i ON s.SiparisID = i.SiparisID;
END
END;
```

### 3-procedure.sql

---puanlama yapan procedure

```
CREATE PROCEDURE PuanlamaYap
    @KullaniciID INT,
    @RestoranID INT,
    @Puan INT,
    @Yorum TEXT
AS
BEGIN
    INSERT INTO Puanlama (KullaniciID, RestoranID, Puan, Yorum)
    VALUES (@KullaniciID, @RestoranID, @Puan, @Yorum)
END
```

--- ödeme procedure

```
CREATE PROCEDURE OdemeYap
    @SiparisID INT,
    @OdemeTuru VARCHAR(50),
    @OdemeDurumu VARCHAR(50)
AS
```

```
BEGIN

    UPDATE Odeme

    SET OdemeTuru = @OdemeTuru, OdemeDurumu = @OdemeDurumu

    WHERE SiparisID = @SiparisID

END
```

---- restoran oluşturma procedure

```
CREATE PROCEDURE RestoranEkleme
```

```
    @RestoranAdi VARCHAR(100),

    @Adres TEXT,

    @Telefon VARCHAR(15),

    @MutfakTuru VARCHAR(50),

    @CalismaSaatleri VARCHAR(100)
```

```
AS
```

```
BEGIN
```

```
    INSERT INTO Restoran (RestoranAdi, Adres, Telefon, MutfakTuru, CalismaSaatleri)

    VALUES

    (@RestoranAdi, @Adres, @Telefon, @MutfakTuru, @CalismaSaatleri);

END;
```

--- Yemek Tablosuna Yeni Veri Ekleme

```
CREATE PROCEDURE YemekEkleme
```

```
    @YemekAdi VARCHAR(100),

    @Aciklama TEXT,

    @Fiyat DECIMAL(10, 2),

    @RestoranID INT
```

AS

BEGIN

INSERT INTO Yemek (YemekAdi, Aciklama, Fiyat, RestoranID)

VALUES

(@YemekAdi, @Aciklama, @Fiyat, @RestoranID);

END;

--- kullanıcı tablosuna veri ekleyen procedure

CREATE PROCEDURE KullanıcıEkleme

@Isim VARCHAR(100),

@Eposta VARCHAR(100),

@Telefon VARCHAR(15),

@Sifre VARCHAR(100),

@Adres TEXT

AS

BEGIN

INSERT INTO Kullanici (Isim, Eposta, Telefon, Sifre, Adres)

VALUES

(@Isim, @Eposta, @Telefon, @Sifre, @Adres);

END;

----Sipariş listeleyen procedure

CREATE PROCEDURE KullaniciSiparisleriniListele

@KullaniciID INT



AS

BEGIN

SET NOCOUNT ON;

SELECT

S.SiparisID,

S.SiparisZamani,

S.TeslimZamani,

S.Durum,

S.ToplamFiyat,

S.TeslimatAdresi

FROM

Siparis AS S

WHERE

S.KullaniciID = @KullaniciID;

IF @@ROWCOUNT = 0

BEGIN

PRINT 'Bu kullanıcıya ait herhangi bir sipariş bulunamadı.';

END

END;

**4-transaction.sql**

---sipariş oluşturma 3 adet tabloda değişim yapıyor

```
CREATE PROCEDURE SiparisOlustur
    @KullaniciID INT,
    @ToplamFiyat DECIMAL(10, 2)
AS
BEGIN
    BEGIN TRANSACTION;

    BEGIN TRY
        DECLARE @TeslimatAdresi VARCHAR(MAX);

        SELECT @TeslimatAdresi = Adres
        FROM Kullanici
        WHERE KullaniciID = @KullaniciID;

        IF @TeslimatAdresi IS NULL
        BEGIN
            PRINT 'Kullanıcı adresi bulunamadı.';
            ROLLBACK;
            RETURN;
        END

        INSERT INTO Siparis (KullaniciID, ToplamFiyat, TeslimatAdresi)
        VALUES (@KullaniciID, @ToplamFiyat, @TeslimatAdresi);

        COMMIT;

        PRINT 'Sipariş oluşturuldu.';
```

```
END TRY

BEGIN CATCH

    ROLLBACK;

    PRINT 'Sipariş oluşturma sırasında sorun oluştu.';

END CATCH

END;
```

----- Teslimat durumu güncellendiğinde oluşabilecek bir hata için transaction.

```
CREATE PROCEDURE TeslimatDurumuGuncelleTransaction
```

```
    @TeslimatID INT,
```

```
    @TeslimatDurumu VARCHAR(50)
```

```
AS
```

```
BEGIN
```

```
    BEGIN TRANSACTION;
```

```
    IF NOT EXISTS (SELECT 1 FROM Teslimat WHERE TeslimatID = @TeslimatID)
```

```
    BEGIN
```

```
        PRINT 'Hata: TeslimatID mevcut değil.';
```

```
        ROLLBACK TRANSACTION;
```

```
        RETURN;
```

```
    END
```

```
BEGIN TRY
```

```
UPDATE Teslimat

SET TeslimatDurumu = @TeslimatDurumu

WHERE TeslimatID = @TeslimatID;


COMMIT TRANSACTION;

END TRY

BEGIN CATCH

    PRINT 'Hata: ' + ERROR_MESSAGE();

    ROLLBACK TRANSACTION;

END CATCH

END;


select * from Teslimat

select * from Kullanici

select * from Siparis


EXEC SiparisOlustur @KullaniciID = 3, @ToplamFiyat =140;


EXEC TeslimatDurumuGuncelleTransaction

    @TeslimatID = 9999,

    @TeslimatDurumu = 'Başarısız';
```

## 5-örnek\_veri.sql

-- Kullanıcılar

INSERT INTO Kullanici (Isim, Eposta, Telefon, Sifre, Adres)

VALUES

('Ali Yılmaz', 'ali@example.com', '5551234567', 'sifre123', 'İstanbul, Türkiye'),

('Ayşe Demir', 'ayse@example.com', '5559876543', 'sifre456', 'Ankara, Türkiye'),

('Mehmet Aslan', 'mehmet@example.com', '5552345678', 'sifre789', 'İzmir, Türkiye'),

('Zeynep Kaya', 'zeynep@example.com', '5553456789', 'sifre321', 'Bursa, Türkiye');

-- Restoranlar

INSERT INTO Restoran (RestoranAdi, Adres, Telefon, MutfakTuru, CalismaSaatleri)

VALUES

('Ali''s Kebap', 'İstanbul, Beyoğlu', '5550001111', 'Kebap', '09:00-22:00'),

('Ayşe''nin Sofrası', 'Ankara, Çankaya', '5552223333', 'Ev Yemekleri', '08:00-20:00'),

('Mehmet''in Mangal', 'İzmir, Karşıyaka', '5553334444', 'Mangal', '10:00-23:00'),

('Zeynep''in Lezzetleri', 'Bursa, Osmangazi', '5554445555', 'Türk Mutfağı', '07:00-21:00');

-- Kategoriler

INSERT INTO Kategori (KategoriAdi)

VALUES

('Kebap'),

('Ev Yemekleri'),

('Mangal'),

('Tatlı'),

('Pasta');

-- Yemekler

INSERT INTO Yemek (YemekAdi, Aciklama, Fiyat, RestoranID)

VALUES

('Adana Kebap', 'Lezzetli baharatlarla hazırlanmış kebab', 50.00, 1),

('Lahmacun', 'Taze yapılmış ince lahmacun', 30.00, 1),

('Kısır', 'Ev yapımı kısır', 20.00, 2),

('Mantı', 'Ev usulü mantı', 35.00, 2),

('Köfte', 'Izgarada pişmiş köfte', 40.00, 3),

('Saray Kebabı', 'Pide ekmeği üzerinde kebab', 60.00, 3),

('Baklava', 'Taze baklava', 25.00, 4),

('Fırın Tavuk', 'Fırınlanmış tavuk ve garnitür', 45.00, 4);

-- Yemek\_Kategori

INSERT INTO Yemek\_Kategori (YemekID, KategoriID)

VALUES

(1, 1), -- Adana Kebap, Kebab

(2, 1), -- Lahmacun, Kebab

(3, 2), -- Kısır, Ev Yemekleri

(4, 2), -- Mantı, Ev Yemekleri

(5, 3), -- Köfte, Mangal

(6, 3), -- Saray Kebabı, Mangal

(7, 4), -- Baklava, Tatlı

(8, 4); -- Fırın Tavuk, Tatlı

-- Siparişler

INSERT INTO Siparis (KullaniciID, Durum, ToplamFiyat, TeslimatAdresi)

VALUES

(1, 'Hazırlanıyor', 80.00, 'İstanbul, Beyoğlu'),  
(2, 'Hazırlanıyor', 70.00, 'Ankara, Çankaya'),  
(3, 'Hazırlanıyor', 100.00, 'İzmir, Karşıyaka'),  
(4, 'Hazırlanıyor', 80.00, 'Bursa, Osmangazi');

-- Sipariş Detayları

INSERT INTO SiparisDetay (SiparisID, YemekID, YemekFiyati, Miktar)

VALUES

(1, 1, 50.00, 1), -- Adana Kebap  
(1, 2, 30.00, 1), -- Lahmacun  
(2, 3, 20.00, 2), -- Kısır  
(2, 4, 35.00, 1), -- Mantı  
(3, 5, 40.00, 2), -- Köfte  
(3, 6, 60.00, 1), -- Saray Kebabı  
(4, 7, 25.00, 3), -- Baklava  
(4, 8, 45.00, 1); -- Fırın Tavuk

-- Ödemeler

INSERT INTO Odeme (SiparisID, OdemeTuru, OdemeDurumu)

VALUES

(1, 'Kredi Kartı', 'Bekliyor'),  
(2, 'Nakit', 'Bekliyor'),  
(3, 'Online', 'Bekliyor'),

```
(4, 'Kredi Kartı', 'Bekliyor');
```

```
-- Puanlamalar
```

```
INSERT INTO Puanlama (KullaniciID, RestoranID, Puan, Yorum)
```

```
VALUES
```

```
(1, 1, 5, 'Harika bir kebab deneyimi!'),
```

```
(2, 2, 4, 'Mantı çok güzeldi, fakat kısır biraz fazla baharatlıydı.'),
```

```
(3, 3, 5, 'Köfte ve Saray Kebabı mükemmeldi!'),
```

```
(4, 4, 4, 'Baklava çok taze ve lezzetliydi, ancak fırın tavuk biraz kuru idi.');
```

```
select * from Kullanici
```

```
6-procedure\_ve\_transaction\_örnek.sql
```

```
----örnek rollback ve commit
```

```
EXEC SiparisOlustur @KullaniciID =5, @ToplamFiyat=170;
```

```
---adresi olmayan bir kullanıcı olduğu için rollback yapacak
```

```
EXEC SiparisOlustur @KullaniciID =3, @ToplamFiyat=170;
```

```
--- bilgiler eksiksiz olduğu için commit edilecek
```

```
EXEC TeslimatDurumuGuncelleTransaction
```

```
@TeslimatID = 9999,
```

```
@TeslimatDurumu = 'Başarısız';
```



---- tabloda olmayan bir ID girildiğinde rollback edilecek

---- örnek procedure komutları

EXEC PuanlamaYap @KullaniciID = 5,@RestoranID = 2, @Puan = 3, @Yorum = 'Lezzetli';

-- Kullanıcı eklemek için prosedürün çalıştırılması

EXEC KullaniciEkleme

@Isim = 'selin',

@Eposta = 'selin@example.com',

@Telefon = '1234567890',

@Sifre = 'sifre123',

@Adres = 'İstanbul, Türkiye';

-- Yemek eklemek için prosedürün çalıştırılması

EXEC YemekEkleme

@YemekAdi = 'Dolma',

@Aciklama = 'Limonlu ve zeytinyagli meze',

@Fiyat = 15.50,

@RestoranID = 1;

-- Restoran eklemek için prosedürün çalıştırılması

EXEC RestoranEkleme

@RestoranAdi = 'Meze Evi',

@Adres = 'Fatih, İstanbul',

@Telefon = '0212-1234567',

@MutfakTuru = 'Türk',

@CalismaSaatleri = '09:00-21:00';

---Sipariş listelemek için procedürün çalıştırılması

EXEC KullaniciSiparisleriniListele @KullaniciID =2

