

## İÇİNDEKİLER

1. Sistem/Gereksinim Analizi ve Kavramsal Modelim Oluşturulması
2. Varlık - İlişki (E-R) modelinin oluşturulması ve tablolaştırılması
3. Tabloların Normalizasyonu
  - 3.1. 'Uye' tablosu için normalizasyon işlemleri
  - 3.2. 'UyelikTipleri' tablosu için normalizasyon işlemleri
  - 3.3. 'Antrenorler' tablosu için normalizasyon işlemleri
  - 3.4. 'AntrenmanProgramlari' tablosu için normalizasyon işlemleri
  - 3.5. 'UyeProgramAtamaları' tablosu için normalizasyon işlemleri
  - 3.6. 'Odemeler' tablosu için normalizasyon işlemleri
  - 3.7. 'Takip' tablosu için normalizasyon işlemleri
  - 3.8. 'Ekipmanlar' tablosu için normalizasyon işlemleri
  - 3.9. 'DiyetPlanlari' tablosu için normalizasyon işlemleri
  - 3.10. 'UyeDiyetAtamaları' tablosu için normalizasyon işlemleri
  - 3.11. 'OdaBilgisi' tablosu için normalizasyon işlemleri
  - 3.12. 'Personel' tablosu için normalizasyon işlemleri
4. İlişkisel Cebir Örnekleri
5. Veri tabanı ve tabloların oluşturulması
  - 5.1. Veri Tabanı Oluşturulması
  - 5.2. Tabloların Oluşturulması
    - 5.2.1. 'UyelikTipleri' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.2. 'Uye' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.3. 'Antrenorler' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.4. 'AntrenmanProgramlari' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.5. 'UyeProgramAtamaları' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.6. 'Odemeler' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.7. 'Takip' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.8. 'Ekipmanlar' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.9. 'DiyetPlanlari' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.10. 'UyeDiyetAtamaları' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.11. 'OdaBilgisi' tablosunun oluşturulması ve verilerin eklenmesi
    - 5.2.12. 'Personel' tablosunun oluşturulması ve verilerin eklenmesi
6. Veri tabanı sorgu örnekleri
  - 6.1. Normal sorgu örnekleri
7. Saklı yordamlar( Stored procedures) örnekleri
8. Tetikleyici (Trigger) örneği

## 1.Sistem/Gereksinim Analizi ve Kavramsal Modelim Oluřturulması

Bir spor salonu yönetiminde kullanılmak üzere bir veri tabanı programlanmak istenmektedir. Veri tabanında; **üyeler, üyelik tipleri, antrenörler, antrenman programları, üye-program atamaları, ödemeler, giriş-çıkış takibi, ekipmanlar, diyet planları, üye-diyet atamaları, salon-oda bilgileri, personel** hakkında veriler tutulacaktır. Ařağıda belirtilen kavramsal model, varlık-ilifki modelindeki ilifkilerin tablolara eklenmesiyle oluřması beklenen řemayı ifade etmektedir.

Her üyeye ait; **üye numarası, ad, soyad, cinsiyet, doğum tarihi, telefon numarası, e-mail adresi, üye olma tarihi, üyelik tipi numarası** bilgileri yer alacaktır. Her üye sadece bir üyelik tipine sahip olabilir. Üye, kendisine atanmış **antrenman programlarını** ve **diyet planlarını** görüntüleyebilir, **giriř-çıkış saatleri** sistemde kayıtlı tutulur. Ayrıca ödeme bilgileri de her üye için ayrı tutulacaktır.

Her üyelik türüne ait; **üyelik tipi numarası, üyelik adı, fiyat ve süresi** (ay olarak) bilgileri yer alacaktır. Her üyelik tipi birden fazla üyeye atanabilir.

Her antrenöre ait: **antrenör numarası, ad, soyad, doğum tarihi, telefon numarası ve uzmanlık alanı** bilgisi yer alacaktır. Antrenörler, belirli antrenman programlarının sorumlusu olabilir. Bir antrenör birden fazla antrenman programına atanabilir.

Her antrenman programına ait; **program numarası, program adı ve açıklama** bilgileri yer alacaktır. Her program bir antrenöre atanır. Üyeler, kendilerine atanan programlar görebilir.

Üye-program atamaları tablosunda; **atama numarası, üye numarası, program numarası, antrenör numarası, başlangıç ve bitiş tarih** bilgisi yer alacaktır. Bir üye birden fazla programa atanabilir.

Ödemeler tablosunda; **ödeme numarası, üye numarası, ödeme tarihi, ödeme tutarı ve ödeme yöntemi (nakit, kredi kartı vb.)** yer alacaktır. Her ödeme yalnızca bir üyeye aittir.

Giriş-çıkış takibi tablosunda; **takip numarası, üye numarası, giriş tarih-saat, çıkış tarih-saat** bilgileri tutulacaktır. Her üyenin birçok giriş-çıkış kaydı olabilir.

Ekipmanlar tablosunda; **ekipman numarası, adı, satın alma tarihi ve ekipman durumu** bilgileri yer alacaktır. Her ekipman bağımsız bir varlık olarak saklanır.

Her diyet planına ait; **diyet plan numarası, plan adı, açıklaması ve günlük kalori miktarı** bilgileri yer alır.

Üye-diyet atamaları tablosunda; **atama numarası, üye numarası, diyet planı numarası, atanma tarihi** bilgileri yer alır. Bir üye birden fazla diyet planı alabilir.

Salon/oda bilgileri tablosunda; **oda numarası, oda adı, kapasitesi ve kullanım amacı** bilgileri yer alacaktır. Oda bilgileri isteğe bağlı olarak programlara veya ekipmanlara bağlanabilir.

Personel tablosunda; **personel numarası, ad, soyad, doğum tarihi, cinsiyet, telefon numarası, bölüm adı, görev türü ve işe başlama tarihi** bilgileri yer alacaktır. Personel, antrenör dışında kalan tüm çalışanları temsil eder.

## **2.Varlık - İlişki (E-R) Modelinin Oluşturulması ve Tablolaştırılması**

Kavramsal modelde belirtilen varlıklara, niteliklere ve ilişkilere uygun olarak modele ait varlık-ilişki (E-R) diyagramı aşağıdaki görselde tasarlanmıştır. Varlıklar arası ilişkiler, 'Crow's foot notation' gösterimine uygun olarak tasarlanmıştır.



### 3.Tabloların Normalizasyonu

Bu proje kapsamında geliştirdiğim spor salonu yönetim sistemi için hazırladığım veri tabanı şeması üzerindeki tüm tabloları, veri tabanı normalizasyon kuralları çerçevesinde (1NF, 2NF ve 3NF) detaylı olarak inceledim. Normalizasyon, veri tabanında veri tekrarını azaltmak, veri tutarlılığını artırmak güncelleme ve silme gibi durumlarda oluşabilecek tutarsızlıkların önüne geçmektedir. Bu nedenle veri tabanımın sağlıklı ve sürdürülebilir bir yapıya sahip olması adına bu adımı önemli buluyorum.

Yaptığım incelemelerde, tabloların her birinin 1NF, 2NF ve 3NF kurallarına uygun şekilde tasarladığımı fark ettim. Her tabloda veriler tekil ve bölünemez şekilde tutuluyor, bu da 1NF'i sağlıyor. Ayrıca tabloların çoğunda tekil birincil anahtar kullanıldığı için kısmi bağımlılık yok, yani 2NF de sağlanıyor. Ek olarak, alanlar arasında transitif bir bağımlılık bulunmadığını da gördüm; bu da 3NF'in sağlandığını gösteriyor. Anahtarları ve ilişkileri tanımlarken mümkün olduğunca sade, anlaşılır ve genişletilebilir bir yapı oluşturmaya çalıştım.

#### 3.1. 'Uye' tablosu için normalizasyon işlemleri

##### 1NF

- Tüm alanlar atomiktir (örneğin ad, soyad, e-posta gibi).
- Sağlanıyor.

##### 2NF

- Anahtar: uyelID → diğer tüm alanlar bu anahtara tam bağımlıdır.
- Sağlanıyor.

##### 3NF

- Tabloda transitif bağımlılık yoktur. Örneğin cinsiyet, e-posta gibi alanlar birbirine değil, doğrudan uyelID'ye bağlıdır.
- Sağlanıyor.

#### 3.2. 'UyelikTipleri' tablosu için normalizasyon işlemleri

##### 1NF

- TipAdi, fiyat, süresi gibi tüm alanlar atomik.
- Sağlanıyor.

##### 2NF

- Anahtar: üyelikTipiID. Diğer tüm alanlar bu anahtara tam bağımlı. Sağlanıyor.

##### 3NF

- Transitif bağımlılık yok. Örneğin fiyat → süre gibi bir ilişki yok.

- Sağlanıyor.

### **3.3. ‘Antrenorler’ tablosu için normalizasyon işlemleri**

1NF

- Ad, soyad, uzmanlık alanı gibi alanlar atomik.
- Sağlanıyor.

2NF

- Anahtar: antrenorID, diğer alanlar bu anahtara tam bağımlıdır.
- Sağlanıyor.

3NF

- Alanlar arasında transitif bağımlılık yok.
- Sağlanıyor.

### **3.4. ‘AntrenmanProgramlari’ tablosu için normalizasyon işlemleri**

1NF

- Program adı, açıklama gibi alanlar atomik.
- Sağlanıyor.

2NF

- Anahtar: programID, tüm alanlar bu anahtara tam bağlı.
- Sağlanıyor.

3NF

- Transitif bağımlılık yok.
- Sağlanıyor.

### **3.5. ‘UyeProgramAtamaları’ tablosu için normalizasyon işlemleri**

1NF

- Başlangıç ve bitiş tarihi gibi alanlar atomik.
- Sağlanıyor.

2NF

- Anahtar: AtamaID, UyeID, ProgramID, AntrenorID bu anahtarla ilişkili.
- Sağlanıyor.

3NF

- Alanlar arasında transitif bağımlılık yok.
- Sağlanıyor.

### **3.6. ‘Odemeler’ tablosu için normalizasyon işlemleri**

1NF

- Ödeme tarihi, tutarı, yöntemi atomiktir.
- Sağlanıyor.

2NF

- Anahtar: OdemeID, diğer alanlar tam bağımlıdır.
- Sağlanıyor.

3NF

- Tabloda transitif bağımlılık gözlenmemektedir.
- Sağlanıyor.

### **3.7. ‘Takip’ tablosu için normalizasyon işlemleri**

1NF

- Giriş/çıkış saatleri atomik.
- Sağlanıyor.

2NF

- Anahtar: Anahtar: TakipID. Diğer bilgiler bu anahtara bağlı. Sağlanıyor.

3NF

- Transitif bağımlılık yoktur.
- Sağlanıyor.

### **3.8. ‘Ekipmanlar’ tablosu için normalizasyon işlemleri**

1NF

- Ekipman adı, durumu gibi bilgiler atomik.
- Sağlanıyor.

2NF

- Anahtar: EkipmanID. Tüm alanlar bu anahtara bağlı.
- Sağlanıyor.

3NF

- Tabloda transitif bağımlılık yoktur.
- Sağlanıyor.

### **3.9. ‘DiyetPlanları’ tablosu için normalizasyon işlemleri**

1NF

- Açıklama, günlük kalori miktarı atomik.
- Sağlanıyor.

2NF

- Anahtar: DiyetPlanID. Tüm alanlar bu anahtara bağlı.
- Sağlanıyor.

3NF

- Tabloda transitif bağımlılık bulunmaz.
- Sağlanıyor.

### **3.10. ‘UyeDiyetAtamaları’ tablosu için normalizasyon işlemleri**

1NF

- Atanma tarihi gibi alanlar atomik.
- Sağlanıyor.

2NF

- Anahtar: UyeDiyetID Diğer alanlar bu anahtara bağlı.
- Sağlanıyor.

3NF

- Tabloda transitif bağımlılık yok.
- Sağlanıyor.

### **3.11. ‘OdaBilgisi’ tablosu için normalizasyon işlemleri**

1NF

- Oda adı, kapasite gibi alanlar atomik.
- Sağlanıyor.

2NF

- Anahtar: OdaID. Diğer alanlar bu anahtara bağlı.
- Sağlanıyor.

3NF

- Transitif bağımlılık gözlenmemekte.
- Sağlanıyor.

### **3.12. ‘Personel’ tablosu için normalizasyon işlemleri**

1NF



- Ad, soyad, görev türü gibi tüm bilgiler atomiktir.
- Sağlanıyor.

2NF

- Anahtar: PersonelID. Diğer tüm alanlar bu anahtara tam bağımlıdır.
- Sağlanıyor.

3NF

- Tabloda transitif bağımlılık yoktur.
- Sağlanıyor.

#### 4.İlişkisel Cebir Örnekleri

Aşağıdaki ilişkisel cebir sorgusu, ‘Kadın’ cinsiyetine sahip üyelerin ad, soyad ve e-posta bilgilerini sorgular.

$\pi$  Ad, Soyad, Eposta ( $\sigma$  Cinsiyet = 'Kadın' (Uye))

Aşağıdaki ilişkisel cebir sorgusu, 01-05-2025 tarihinde yapılan ödemelere ait ödeme tutarı ve ödeme yöntemi bilgilerini sorgular.

$\pi$  Tutar, OdemeYontemi ( $\sigma$  OdemeTarihi = '2025-05-01' (Odemeler))

Aşağıdaki ilişkisel cebir sorgusu, üyelik süresi 12 aydan fazla olan üyelik tiplerinin ad ve fiyat bilgilerini sorgular.

$\pi$  TipAdi, Ucret ( $\sigma$  SureAy > 12 (UyelikTipleri))

Aşağıdaki ilişkisel cebir sorgusu, 500 kaloriden fazla enerji içeren diyet planlarının adlarını ve kalori miktarlarını listeler.

$\pi$  PlanAdi, GunlukKalori ( $\sigma$  GunlukKalori > 500 (DiyetPlanlari))

Aşağıdaki ilişkisel cebir sorgusu, 22-05-2024 tarihinde yapılan ve puanı 5'ten büyük olan giriş-çıkış kayıtları bilgilerini verir.

$\sigma$  Giris = '2024-05-22'  $\wedge$  Puan > 5 (Takip)

Aşağıdaki ilişkisel cebir sorgusu, birden fazla antrenman programı atanmış üyelerin ad ve soyad bilgilerini sorgular.

$\pi$  Ad, Soyad (

(Uye  $\bowtie$  Uye.UyeID = UyeProgramAtamalari.UyeID)

GrupBy Uye.UyeID Having COUNT(ProgramID) > 1)

Aşağıdaki ilişkisel cebir sorgusu, 'Yoga' programına atanmış üyelerin ad, soyad ve atanma tarihlerini sorgular.

$\pi$  Ad, Soyad, BaslangicTarihi (

( $\sigma$  ProgramAdi = 'Yoga' (AntrenmanProgramlari)

$\bowtie$  AntrenmanProgramlari.ProgramID =  
UyeProgramAtamalari.ProgramID)

$\bowtie$  UyeProgramAtamalari.UyeID = Uye.UyeID)

Aşağıdaki ilişkisel cebir sorgusu, 'Yoga' programına atanmış üyelerin ad, soyad ve atanma tarihlerini sorgular.

$\pi$  Ad, Soyad, BaslangicTarihi (

( $\sigma$  ProgramAdi = 'Yoga' (AntrenmanProgramlari)

$\bowtie$  AntrenmanProgramlari.ProgramID = UyeProgramAtamalari.ProgramID)

$\bowtie$  UyeProgramAtamalari.UyeID = Uye.UyeID)

Aşağıdaki ilişkisel cebir sorgusu, 'Kondisyon Programı' adlı antrenman programına atanan üyelerin ad ve soyad bilgilerini getirir.

$\pi$  Ad, Soyad (

$\sigma$  ProgramAdi = 'Kondisyon Programı' (AntrenmanProgramlari)

$\bowtie$  AntrenmanProgramlari.ProgramID = UyeProgramAtamalari.ProgramID)

$\bowtie$  UyeProgramAtamalari.UyeID = Uye.UyeID)

Aşağıdaki ilişkisel cebir sorgusu, 'Yoga' uzmanlık alanına sahip antrenörlerin verdiği programların adlarını verir.

$\pi$  ProgramAdi (

$\sigma$  Uzmanlik = 'Yoga' (

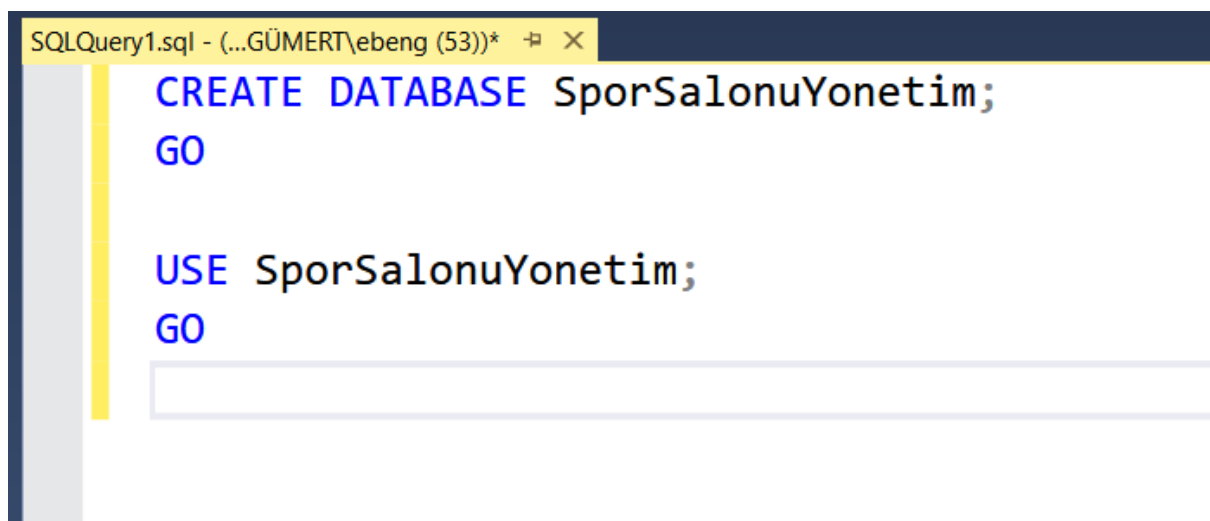
Antrenorler  $\bowtie$  Antrenorler.AntrenorID = UyeProgramAtamalari.AntrenorID

$\bowtie$  UyeProgramAtamalari.ProgramID = AntrenmanProgramlari.ProgramID))

## 5. Veri Tabanı ve Tabloların Oluşturulması

### 5.1. Veri tabanı oluşturulması

CREATE komutu ile 'SporSalonuYonetim' isminde bir veri tabanı oluşturuyoruz. Ardından 'USE SporSalonuYonetim' ve 'GO' komutları ile veri tabanını seçerek sonraki işlemler için kullanılmaya hazır hâle getiriyoruz.



```
SQLQuery1.sql - (...GÜMERT\ebeng (53))*  
CREATE DATABASE SporSalonuYonetim;  
GO  
  
USE SporSalonuYonetim;  
GO
```

Şekil-3: Veri tabanının oluşturulmasıyla ilgili SQL komutları

## 5.2. Tabloların Oluşturulması

### 5.2.1. 'UyelelikTipleri' tablosunun oluşturulması ve verilerin eklenmesi

- **UyelikTipiID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **TipAdi:** Maksimum 50 karakter uzunluğundadır. (örn: "Aylık", "Yıllık")
- **Ucret:** Ondalıklı sayı (DECIMAL(10,2)), üyelik ücretini temsil eder.
- **SureAy:** Tamsayıdır. Üyelik süresi ay cinsindendir.

```
CREATE TABLE UyelikTipleri (  
    UyelikTipiID INT PRIMARY KEY IDENTITY,  
    TipAdi VARCHAR(50),  
    Ucret DECIMAL(10,2),  
    SureAy INT  
);
```

Şekil-4: 'UyelelikTipleri' tablosunun oluşturulmasıyla ilgili SQL komutları

```
INSERT INTO UyelikTipleri (TipAdi, Ucret, SureAy)  
VALUES  
( 'Aylık Üyelik', 200.00, 1),  
( 'Yıllık Üyelik', 2200.00, 12),  
( '3 Aylık Üyelik', 600.00, 3),  
( '6 Aylık Üyelik', 1200.00, 6),  
( 'Öğrenci Üyeliği', 150.00, 1),  
( 'VIP Üyelik', 400.00, 1),  
( 'Grup Üyeliği', 1800.00, 12),  
( 'Bireysel Üyelik', 300.00, 1),  
( 'Kadın Üyeliği', 250.00, 1),  
( 'Erkek Üyeliği', 250.00, 1);
```

Şekil-5: 'UyelelikTipleri' tablosuna örnek verilerin eklenmesi

### 5.2.2. 'Uye' tablosunun oluşturulması ve verilerin eklenmesi

```
CREATE TABLE Uye (
    UyeID INT PRIMARY KEY IDENTITY,
    Ad VARCHAR(50),
    Soyad VARCHAR(50),
    Cinsiyet VARCHAR(10),
    DogumTarihi DATE,
    Telefon VARCHAR(15),
    Eposta VARCHAR(100),
    KayitTarihi DATE,
    UyelikTipiID INT,
    FOREIGN KEY (UyelikTipiID) REFERENCES UyelikTipleri(UyelikTipiID)
);
```

- **UyeID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **Ad:** Maksimum 50 karakter uzunluğundadır.
- **Soyad:** Maksimum 50 karakter uzunluğundadır.
- **Cinsiyet:** Maksimum 10 karakterden oluşur. (örn: “Erkek”, “Kadın”)
- **DogumTarihi:** DATE formatında tarih girilir.
- **Telefon:** Maksimum 15 karakter uzunluğundadır.
- **Eposta:** Maksimum 100 karakter uzunluğundadır.
- **KayitTarihi:** DATE formatında, üye giriş tarihi belirtilir.
- **UyelikTipiID:** Yabancı anahtar. UyelikTipleri tablosuna referans verir.

**Şekil-6: ‘Uye’ tablosunun oluşturulmasıyla ilgili SQL komutları**

```
INSERT INTO Uye (Ad, Soyad, Cinsiyet, DogumTarihi, Telefon, Eposta, KayitTarihi, UyelikTipiID)
VALUES
('Efe', 'Küçük', 'Erkek', '1995-04-12', '5558765432', 'efe@example.com', '2025-04-20', 1),
('Lara', 'Yılmaz', 'Kadın', '1998-07-20', '5559876543', 'lara@example.com', '2025-04-22', 2),
('Berk', 'Sarı', 'Erkek', '1994-11-15', '5556543210', 'berk@example.com', '2025-04-21', 1),
('Merve', 'Aydın', 'Kadın', '2000-01-10', '5553219876', 'merve@example.com', '2025-04-23', 2),
('Serkan', 'Demir', 'Erkek', '1990-03-22', '5555678901', 'serkan@example.com', '2025-04-19', 3),
('Nisan', 'Güler', 'Kadın', '1997-05-17', '5552348901', 'nisan@example.com', '2025-04-24', 1),
('Tuna', 'Çelik', 'Erkek', '1989-12-09', '5558765432', 'tuna@example.com', '2025-04-18', 2),
('Selin', 'Kaya', 'Kadın', '1993-06-30', '5556541234', 'selin@example.com', '2025-04-15', 1),
('Kadir', 'Aydın', 'Erkek', '1992-10-05', '5554321098', 'kadir@example.com', '2025-04-14', 3),
('Elif', 'Kurt', 'Kadın', '1996-04-22', '5559876543', 'elif@example.com', '2025-04-17', 2);
```

**Şekil-7: ‘Uye’ tablosuna örnek verilerin eklenmesi**

### 5.2.3. ‘Antrenorler’ tablosunun oluşturulması ve verilerin eklenmesi

- **AntrenorID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **Ad:** Maksimum 50 karakter uzunluğundadır.
- **Soyad:** Maksimum 50 karakter uzunluğundadır.
- **Telefon:** Maksimum 15 karakter uzunluğundadır.
- **Uzmanlik:** Maksimum 100 karakter uzunluğundadır. (örn: “Kardiyo”, “Ağırlık Antrenmanı”)

```

query4.sql - (...GÜMERT\ebeng (77))*  X  SQLQuery3.sql - (...GÜMERT\ebeng (75))*  SQLQu
CREATE TABLE Antrenorler (
    AntrenorID INT PRIMARY KEY IDENTITY,
    Ad VARCHAR(50),
    Soyad VARCHAR(50),
    Telefon VARCHAR(15),
    Uzmanlik VARCHAR(100)
);

```

Şekil-8: ‘Antrenorler’ tablosunun oluşturulmasıyla ilgili SQL komutları

```

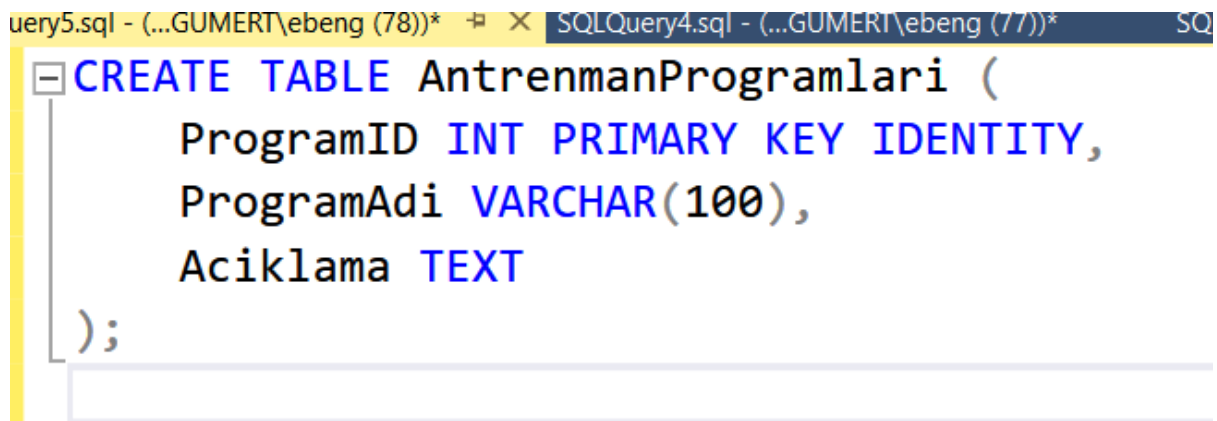
INSERT INTO Antrenorler (Ad, Soyad, Telefon, Uzmanlik)
VALUES
('Ali', 'Kaya', '5556789012', 'Fitness ve Kardiyo'),
('Zeynep', 'Güler', '5552348901', 'Ağırsiklet Antrenörü'),
('Murat', 'Yılmaz', '5551234567', 'Yoga ve Meditasyon Uzmanı'),
('Fatma', 'Özdemir', '5553456789', 'Pilates Eğitmeni'),
('Ahmet', 'Kılıç', '5554567890', 'Boks Antrenörü'),
('Selin', 'Çetin', '5555678901', 'Gruplu Egzersiz Eğitmeni'),
('Emre', 'Demir', '5556789012', 'Kardiyo Antrenörü'),
('Canan', 'Duman', '5557890123', 'Vegan Diyet Uzmanı'),
('Zeynep', 'Aydın', '5558901234', 'Zumba Eğitmeni'),
('Berk', 'Sarı', '5559012345', 'Kas Yapma Uzmanı');

```

Şekil-9: ‘Antrenorler’ tablosuna örnek verilerin eklenmesi

#### 5.2.4. 'AntrenmanProgramlari' tablosunun oluşturulması ve verilerin eklenmesi

- **ProgramID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **ProgramAdi:** Maksimum 100 karakter uzunluğundadır.
- **Aciklama:** Serbest metin (TEXT) türündedir. Programın detayları girilir.

The image shows a screenshot of a SQL query editor window. The title bar indicates the file is 'SQLQuery4.sql'. The editor contains a SQL command to create a table named 'AntrenmanProgramlari'. The command is: 'CREATE TABLE AntrenmanProgramlari ( ProgramID INT PRIMARY KEY IDENTITY, ProgramAdi VARCHAR(100), Aciklama TEXT );'. The text is color-coded: 'CREATE TABLE' is blue, 'AntrenmanProgramlari' is black, and the column definitions are in blue. The window has a yellow sidebar on the left and a blue title bar.

```
CREATE TABLE AntrenmanProgramlari (  
    ProgramID INT PRIMARY KEY IDENTITY,  
    ProgramAdi VARCHAR(100),  
    Aciklama TEXT  
);
```

Şekil-10: 'AntrenmanProgramlari' tablosunun oluşturulmasıyla ilgili SQL komutları

```

INSERT INTO AntrenmanProgramlari (ProgramAdi, Aciklama)
VALUES
('Kilo Verme Programı', 'Haftada 4 gün, kardiyo ve ağırsiklet kombinasyonu'),
('Kas Yapma Programı', 'Haftada 5 gün, ağırsiklet ve beslenme planı'),
('Detoks Programı', 'Vücutta birikmiş toksinleri atmak için detoks programı'),
('Kardiyo Programı', 'Haftada 3 gün, düşük yoğunluklu kardiyo programı'),
('Vegan Diyet Programı', 'Vegan beslenme planı ile zayıflama programı'),
('Paleo Diyet Programı', 'Paleo beslenme planına dayalı kilo verme programı'),
('Ketojenik Program', 'Ketojenik diyet ve egzersiz programı'),
('Yüksek Protein Programı', 'Kas yapmaya yönelik yüksek protein ağırlıklı program'),
('Hızlı Kilo Kaybı Programı', 'Düşük kalorili hızlı kilo kaybı programı'),
('Zumba Programı', 'Dans temelli eğlenceli bir kardiyo programı');

```

**Şekil-11: ‘AntrenmanProgramlari’ tablosuna örnek verilerin eklenmesi**

### 5.2.5. ‘UyeProgramAtamalari’ tablosunun oluşturulması ve verilerin eklenmesi

- **UyeProgramID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **UyeID:** Yabancı anahtar. Uye tablosuna referans verir.
- **ProgramID:** Yabancı anahtar. AntrenmanProgramlari tablosuna referans verir.
- **AntrenorID:** Yabancı anahtar. Antrenorler tablosuna referans verir.
- **BaslangicTarihi:** Başlangıç tarihi, DATE formatında girilir.
- **BitisTarihi:** Bitiş tarihi, DATE formatında girilir.

```

CREATE TABLE UyeProgramAtamalari (
    UyeProgramID INT PRIMARY KEY IDENTITY,
    UyeID INT,
    ProgramID INT,
    AntrenorID INT,
    BaslangicTarihi DATE,
    BitisTarihi DATE,
    FOREIGN KEY (UyeID) REFERENCES Uye(UyeID),
    FOREIGN KEY (ProgramID) REFERENCES AntrenmanProgramlari(ProgramID),
    FOREIGN KEY (AntrenorID) REFERENCES Antrenorler(AntrenorID)
);

```

**Şekil-12: ‘UyeProgramAtamalari’ tablosunun oluşturulmasıyla ilgili SQL komutları**



```
SQLQuery18.sql - ...GUMERT\ebeng (94))* SQLQuery17.sql - ...GUMERT\ebeng (70))* SQLQuery16.sql - ...GUMERT\ebeng (69))* SQLQuery15.sql - ...GUMERT\ebeng (85))*
INSERT INTO UyeProgramAtamaları (UyeID, ProgramID, AntrenorID, BaslangicTarihi, BitisTarihi)
VALUES
(1, 1, 1, '2025-05-01', '2025-06-01'),
(2, 2, 2, '2025-05-03', '2025-06-03'),
(3, 3, 1, '2025-05-02', '2025-06-02'),
(4, 1, 2, '2025-05-01', '2025-06-01'),
(5, 2, 1, '2025-05-03', '2025-06-03'),
(6, 3, 2, '2025-05-02', '2025-06-02'),
(7, 1, 1, '2025-05-01', '2025-06-01'),
(8, 2, 2, '2025-05-03', '2025-06-03'),
(9, 3, 1, '2025-05-02', '2025-06-02'),
(10, 1, 2, '2025-05-01', '2025-06-01');
```

Şekil-13: 'UyeProgramAtamaları' tablosuna örnek verilerin eklenmesi

### 5.2.6. 'Odemeler' tablosunun oluşturulması ve verilerin eklenmesi

- **OdemeID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **UyeID:** Yabancı anahtardır. Uye tablosuna referans verir.
- **Tutar:** Ondalıklı sayıdır (DECIMAL(10,2)), ödeme tutarını temsil eder.
- **OdemeTarihi:** DATE türündedir, ödemenin yapıldığı tarih girilir.
- **OdemeYontemi:** Maksimum 50 karakter uzunluğundadır. (örn: Nakit, Kredi Kartı)

```
CREATE TABLE Odemeler (
    OdemeID INT PRIMARY KEY IDENTITY,
    UyeID INT,
    Tutar DECIMAL(10,2),
    OdemeTarihi DATE,
    OdemeYontemi VARCHAR(50),
    FOREIGN KEY (UyeID) REFERENCES Uye(UyeID)
);
```

Şekil-14: 'Odemeler' tablosunun oluşturulmasıyla ilgili SQL komutları

```
INSERT INTO Odemeler (UyeID, Tutar, OdemeTarihi, OdemeYontemi)  
VALUES  
(1, 300.00, '2025-05-01', 'Kredi Kartı'),  
(2, 250.00, '2025-05-02', 'Nakit'),  
(3, 220.00, '2025-05-03', 'Kredi Kartı'),  
(4, 200.00, '2025-05-01', 'Nakit'),  
(5, 350.00, '2025-05-02', 'Kredi Kartı'),  
(6, 280.00, '2025-05-03', 'Nakit'),  
(7, 300.00, '2025-05-01', 'Kredi Kartı'),  
(8, 250.00, '2025-05-02', 'Kredi Kartı'),  
(9, 270.00, '2025-05-03', 'Nakit'),  
(10, 320.00, '2025-05-02', 'Kredi Kartı');
```

Şekil-15: 'Odemeler' tablosuna örnek verilerin eklenmesi

#### 5.2.7. 'Takip' tablosunun oluşturulması ve verilerin eklenmesi

- **TakipID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **UyeID:** Yabancı anahtardır. Uye tablosundaki UyeID değerine referans verir.
- **Giris:** Tarih ve saat (DATETIME) türünde, üyenin giriş zamanı.
- **Cikis:** Tarih ve saat (DATETIME) türünde, üyenin çıkış zamanı.

```
SQLQuery8.sql - (...GÜMERT\ebeng (62))*  SQLQuery7.sql - (...GÜMERT\ebeng (54))*  SQLQuery6.s  
CREATE TABLE Takip (  
    TakipID INT PRIMARY KEY IDENTITY,  
    UyeID INT,  
    Giris DATETIME,  
    Cikis DATETIME,  
    FOREIGN KEY (UyeID) REFERENCES Uye(UyeID)  
);
```

Şekil-16: 'Takip' tablosunun oluşturulmasıyla ilgili SQL komutları

```

INSERT INTO Takip (UyeID, Giris, Cikis)
VALUES
(1, '2025-05-01 09:00:00', '2025-05-01 10:00:00'),
(2, '2025-05-02 08:30:00', '2025-05-02 09:30:00'),
(3, '2025-05-01 10:00:00', '2025-05-01 11:00:00'),
(4, '2025-05-02 09:30:00', '2025-05-02 10:30:00'),
(5, '2025-05-03 10:15:00', '2025-05-03 11:15:00'),
(6, '2025-05-03 08:00:00', '2025-05-03 09:00:00'),
(7, '2025-05-01 11:00:00', '2025-05-01 12:00:00'),
(8, '2025-05-01 14:00:00', '2025-05-01 15:00:00'),
(9, '2025-05-02 07:30:00', '2025-05-02 08:30:00'),
(10, '2025-05-02 16:00:00', '2025-05-02 17:00:00');

```

Şekil-17: 'Takip' tablosuna örnek verilerin eklenmesi

#### 5.2.8. 'Ekipmanlar' tablosunun oluşturulması ve verilerin eklenmesi

- **EkipmanID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **Adi:** Maksimum 100 karakter uzunluğundadır.
- **SatinAlmaTarihi:** Satın alma tarihi, YYYY-MM-DD formatında girilir.
- **Durum:** Maksimum 50 karakter uzunluğundadır. Ekipmanın durumu açıklanır (örneğin: "Kullanımda", "Bakımda").

```

CREATE TABLE Ekipmanlar (
    EkipmanID INT PRIMARY KEY IDENTITY,
    Adi VARCHAR(100),
    SatinAlmaTarihi DATE,
    Durum VARCHAR(50)
);

```

Şekil-18: 'Ekipmanlar' tablosunun oluşturulmasıyla ilgili SQL komutları

```
INSERT INTO Ekipmanlar (Adi, SatinAlmaTarihi, Durum)
VALUES
('Dambıl', '2022-03-10', 'İyi'),
('Koşu Bandı', '2023-02-14', 'Yeni'),
('Boks Torbası', '2021-08-05', 'Kullanıma Uygun'),
('Bisiklet Ergometresi', '2020-11-20', 'İyi'),
('Ağırsiklet', '2022-04-18', 'Yeni'),
('Yogamat', '2021-05-15', 'İyi'),
('Pilates Topu', '2022-08-01', 'Yeni'),
('Zumba Step Tahtası', '2020-07-25', 'Kullanıma Uygun'),
('Eliptik Bisiklet', '2023-01-30', 'İyi'),
('Kettlebell', '2022-09-18', 'Yeni');
```

Şekil-19: 'Ekipmanlar' tablosuna örnek verilerin eklenmesi

#### 5.2.9. 'DiyetPlanlari' tablosunun oluşturulması ve verilerin eklenmesi

- **DiyetPlanID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **PlanAdi:** Maksimum 100 karakterden oluşur.
- **Aciklama:** Serbest metin (TEXT) alanıdır. Planın ayrıntılı açıklaması yazılır.
- **GunlukKalori:** Tamsayıdır. Günlük önerilen kalori miktarını içerir.

```
CREATE TABLE DiyetPlanlari (
    DiyetPlanID INT PRIMARY KEY IDENTITY,
    PlanAdi VARCHAR(100),
    Aciklama TEXT,
    GunlukKalori INT
);
```

Şekil-20: 'DiyetPlanlari' tablosunun oluşturulmasıyla ilgili SQL komutları

```

INSERT INTO DiyetPlanlari (PlanAdi, Aciklama, GunlukKalori)
VALUES
('Kilo Verme Planı', 'Haftada 1 kg vermek için düşük kalorili diyet planı', 1500),
('Kas Yapma Planı', 'Ağırsiklet antrenmanları için kas yapma odaklı diyet', 2500),
('Detoks Planı', 'Vücutta birikmiş toksinleri atmak için detoks diyeti', 1800),
('Kardiyo Planı', 'Kalp sağlığı için düşük kalorili kardiyo odaklı diyet', 2000),
('Vegan Diyet', 'Hayvansal ürünlerden uzak durarak sağlıklı vegan beslenme planı', 2200),
('Ketojenik Diyet', 'Yüksek yağ, düşük karbonhidrat diyeti', 1800),
('Paleo Diyeti', 'Doğal gıdalarla zenginleştirilmiş paleo diyeti', 2300),
('Hızlı Kilo Kaybı Planı', 'Kısa sürede hızlı kilo kaybı sağlamak için düşük kalorili diyet', 1300),
('Yüksek Protein Diyeti', 'Kas geliştirme amacıyla yüksek protein diyeti', 2700),
('Düşük Glisemik İndeks Diyeti', 'Kan şekeri seviyesini kontrol altına almak için düşük glisemik indeks diyeti', 1900);

```

Şekil-21: ‘DiyetPlanlari’ tablosuna örnek verilerin eklenmesi

#### 5.2.10. ‘UyeDiyetAtamalari’ tablosunun oluşturulması ve verilerin eklenmesi

- **UyeDiyetID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **UyeID :** Yabancı anahtardır. Uye tablosundaki UyeID alanına referans verir. Atamanın hangi üyeye ait olduğunu belirtir. Tam sayı türündedir (INT).
- **DiyetPlanID:** Yabancı anahtardır. DiyetPlanlari tablosundaki DiyetPlanID alanına referans verir. Atanan diyet planını belirtir. Tam sayı türündedir (INT).
- **AtamaTarihi:** DATE veri tipindedir. Diyet planının atandığı tarihi belirtir. YYYY-MM-DD formatında tarih değeri alır.

```

CREATE TABLE UyeDiyetAtamalari (
    UyeDiyetID INT PRIMARY KEY IDENTITY,
    UyeID INT,
    DiyetPlanID INT,
    AtamaTarihi DATE,
    FOREIGN KEY (UyeID) REFERENCES Uye(UyeID),
    FOREIGN KEY (DiyetPlanID) REFERENCES DiyetPlanlari(DiyetPlanID)
);

```

Şekil-22: ‘UyeDiyetAtamalari’ tablosunun oluşturulmasıyla ilgili SQL komutları

```

INSERT INTO UyeDiyetAtamaları (UyeID, DiyetPlanID, AtamaTarihi) VALUES
(1, 2, '2024-07-05'),
(2, 3, '2024-07-12'),
(3, 1, '2024-08-01'),
(4, 4, '2024-08-15'),
(5, 5, '2024-09-01'),
(6, 2, '2024-09-10'),
(7, 3, '2024-09-25'),
(8, 1, '2024-10-05'),
(9, 4, '2024-10-20'),
(10, 5, '2024-11-01');

```

Şekil-23: 'UyeDiyetAtamaları' tablosuna örnek verilerin eklenmesi

#### 5.2.11. 'OdaBilgisi' tablosunun oluşturulması ve verilerin eklenmesi

- **OdaID:** Birincil anahtar. Otomatik artan tam sayı değeridir.
- **OdaAdi:** Maksimum 50 karakter uzunluğundadır.
- **Kapasite:** Tamsayı türündedir. Odaya maksimum alınabilecek kişi sayısını belirtir.
- **Amac:** Maksimum 100 karakter uzunluğundadır. Odanın kullanım amacı açıklanır (örneğin: "Yoga Salonu", "Ağırlık Odası").

```

CREATE TABLE OdaBilgisi (
    OdaID INT PRIMARY KEY IDENTITY,
    OdaAdi VARCHAR(50),
    Kapasite INT,
    Amac VARCHAR(100)
);

```

Şekil-24: 'OdaBilgisi' tablosunun oluşturulmasıyla ilgili SQL komutları

```
INSERT INTO OdaBilgisi (OdaAdi, Kapasite, Amac)
VALUES
('Yoga Salonu', 20, 'Yoga ve meditasyon seansları'),
('Fitness Salonu', 30, 'Ağırsiklet ve kardiyo'),
('Boks Salonu', 15, 'Boks antrenmanları'),
('Pilates Salonu', 12, 'Pilates seansları'),
('Zumba Salonu', 25, 'Zumba ve dans dersleri'),
('Koşu Parkuru', 10, 'Koşu ve yürüyüş parkuru'),
('Ağırsiklet Salonu', 18, 'Ağırsiklet ve kuvvet antrenmanları'),
('Gruplu Egzersiz Salonu', 30, 'Grup fitness dersleri'),
('Dinlenme Salonu', 8, 'Dinlenme ve rahatlama alanı'),
('Sauna ve Buhar Odası', 5, 'Sauna ve rahatlama odası');
```

Şekil-25: ‘OdaBilgisi’ tablosuna örnek verilerin eklenmesi

#### 5.2.12. ‘Personel’ tablosunun oluşturulması ve verilerin eklenmesi

- **PersonelID:** Birincil anahtar olarak tanımlanmıştır. Otomatik artan tam sayı değeridir.
- **Ad:** Maksimum 50 karakter uzunluğundadır.
- **Soyad:** Maksimum 50 karakter uzunluğundadır.
- **Gorev:** Maksimum 50 karakter uzunluğundadır. *Antrenör* ve *Eğitmen* değerleri bu tabloda kullanılmaz.
- **Telefon:** Maksimum 15 karakterden oluşur, telefon numarasını metin olarak saklar.
- **IseAlimTarihi:** Yıl-ay-gün (YYYY-MM-DD) biçiminde tarih verisi alır.

```
CREATE TABLE Personel (  
    PersonelID INT PRIMARY KEY IDENTITY,  
    Ad VARCHAR(50),  
    Soyad VARCHAR(50),  
    Gorev VARCHAR(50),  
    Telefon VARCHAR(15),  
    IseAlimTarihi DATE  
);
```

Şekil-26: 'Personel' tablosunun oluşturulmasıyla ilgili SQL komutları

```
INSERT INTO Personel (Ad, Soyad, Gorev, Telefon, IseAlimTarihi)  
VALUES  
( 'Ahmet', 'Yılmaz', 'Antrenör', '5551234567', '2023-01-15'),  
( 'Ayşe', 'Demir', 'Yönetici', '5552345678', '2022-05-20'),  
( 'Mehmet', 'Çelik', 'Temizlik Görevlisi', '5553456789', '2021-11-10'),  
( 'Fatma', 'Gül', 'Antrenör', '5554567890', '2023-03-01'),  
( 'Emre', 'Kaya', 'Yönetici', '5555678901', '2021-06-15'),  
( 'Zeynep', 'Aydın', 'Eğitmen', '5556789012', '2020-09-10'),  
( 'Ali', 'Öztürk', 'Yönetici', '5557890123', '2019-08-22'),  
( 'Berk', 'Sarı', 'Antrenör', '5558901234', '2022-12-10'),  
( 'Canan', 'Kurt', 'Temizlik Görevlisi', '5559012345', '2021-02-05'),  
( 'Veli', 'Şahin', 'Güvenlik Görevlisi', '5550123456', '2020-01-15');
```

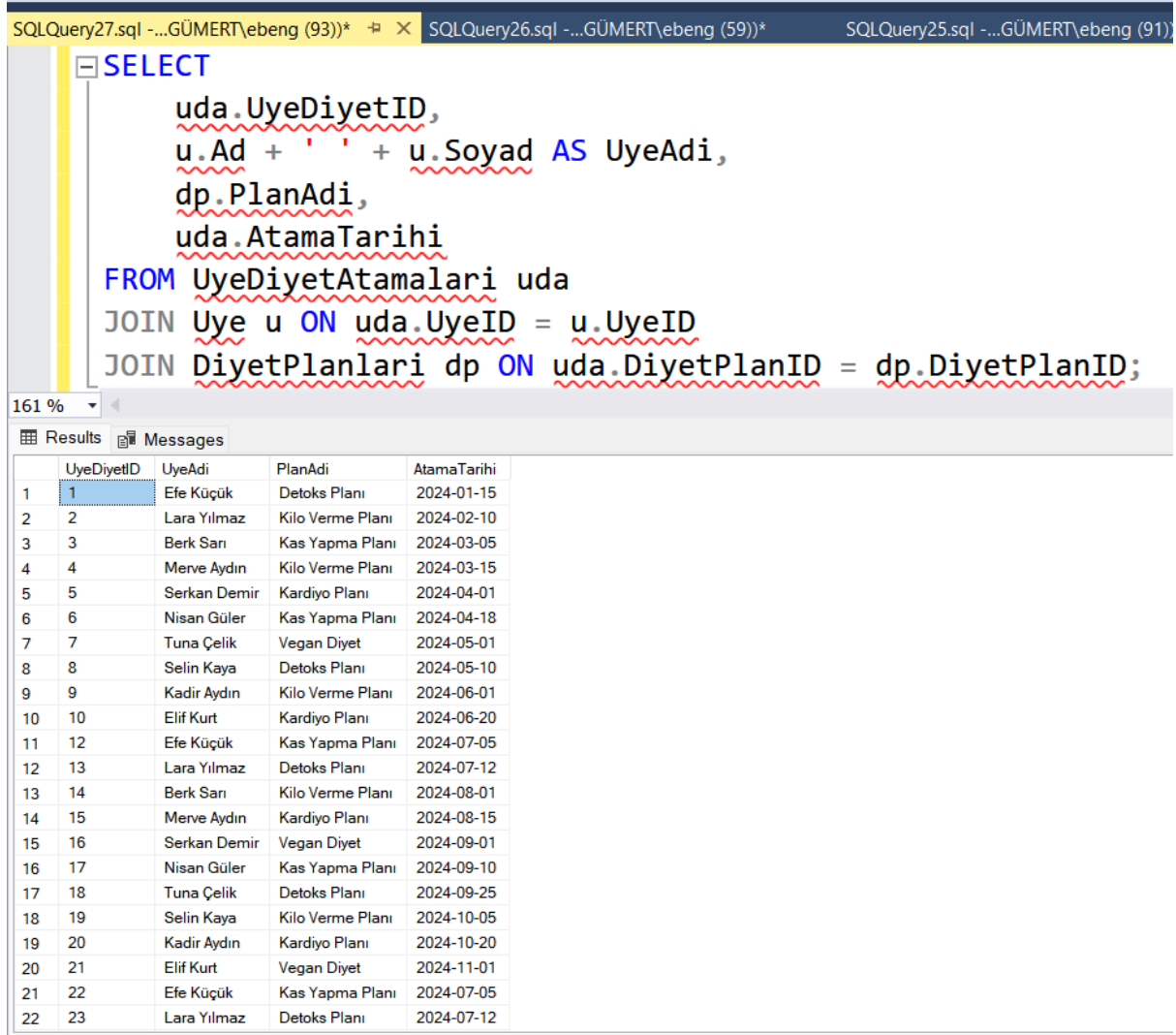
Şekil-27: 'Personel' tablosuna örnek verilerin eklenmesi

## 6. Veri tabanı sorgu örnekleri

### 6.1. Normal sorgu örnekleri



Aşağıdaki SQL sorgusunda, UyeDiyetAtamaları tablosu ile Uye ve DiyetPlanlari tabloları birleştirilmiş, her üyeye ait diyet planı atamaları detaylı olarak listelenmiştir. Bu sayede hangi üyenin hangi diyet planına ne zaman atandığı bilgisi elde edilmiştir.

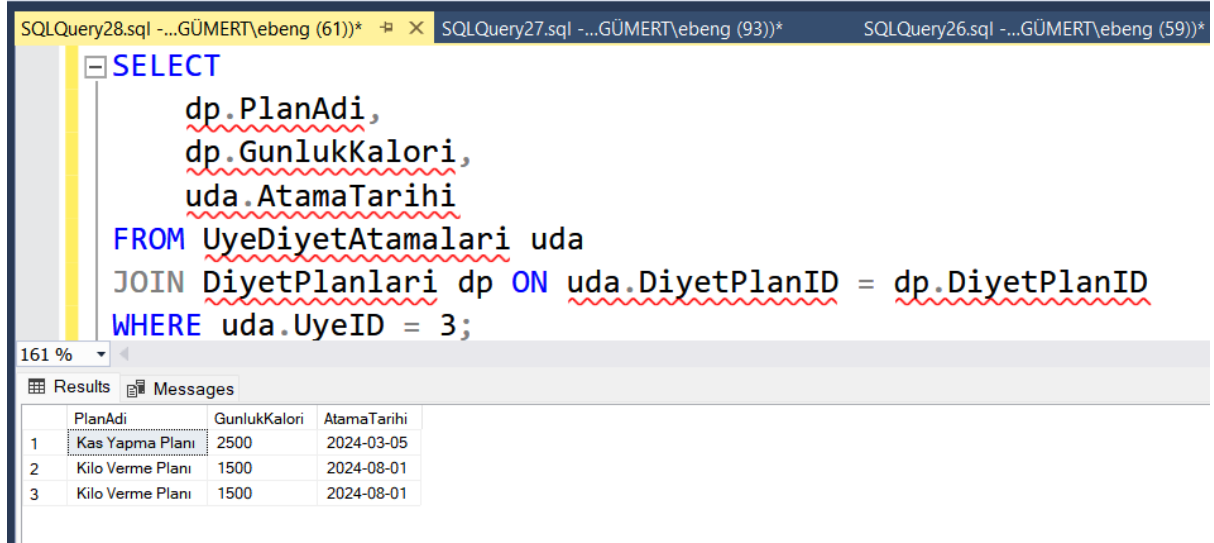


```
SELECT
    uda.UyeDiyetID,
    u.Ad + ' ' + u.Soyad AS UyeAdi,
    dp.PlanAdi,
    uda.AtamaTarihi
FROM UyeDiyetAtamaları uda
JOIN Uye u ON uda.UyeID = u.UyeID
JOIN DiyetPlanlari dp ON uda.DiyetPlanID = dp.DiyetPlanID;
```

	UyeDiyetID	UyeAdi	PlanAdi	AtamaTarihi
1	1	Efe Küçük	Detoks Planı	2024-01-15
2	2	Lara Yılmaz	Kilo Verme Planı	2024-02-10
3	3	Berk Sarı	Kas Yapma Planı	2024-03-05
4	4	Merve Aydın	Kilo Verme Planı	2024-03-15
5	5	Serkan Demir	Kardiyo Planı	2024-04-01
6	6	Nisan Güler	Kas Yapma Planı	2024-04-18
7	7	Tuna Çelik	Vegan Diyet	2024-05-01
8	8	Selin Kaya	Detoks Planı	2024-05-10
9	9	Kadir Aydın	Kilo Verme Planı	2024-06-01
10	10	Elif Kurt	Kardiyo Planı	2024-06-20
11	12	Efe Küçük	Kas Yapma Planı	2024-07-05
12	13	Lara Yılmaz	Detoks Planı	2024-07-12
13	14	Berk Sarı	Kilo Verme Planı	2024-08-01
14	15	Merve Aydın	Kardiyo Planı	2024-08-15
15	16	Serkan Demir	Vegan Diyet	2024-09-01
16	17	Nisan Güler	Kas Yapma Planı	2024-09-10
17	18	Tuna Çelik	Detoks Planı	2024-09-25
18	19	Selin Kaya	Kilo Verme Planı	2024-10-05
19	20	Kadir Aydın	Kardiyo Planı	2024-10-20
20	21	Elif Kurt	Vegan Diyet	2024-11-01
21	22	Efe Küçük	Kas Yapma Planı	2024-07-05
22	23	Lara Yılmaz	Detoks Planı	2024-07-12

Şekil-28: Üyelere ait diyet atamalarını gösteren SQL komutu

Aşağıdaki sorgu ile belirli bir üyeye (örneğin UyeID = 3) atanan tüm diyet planları, plan adı ve günlük kalori bilgisiyle birlikte listelenmiştir. Böylece üyeye özel plan geçmişi görüntülenmiştir.



The screenshot shows a SQL query window with the following query:

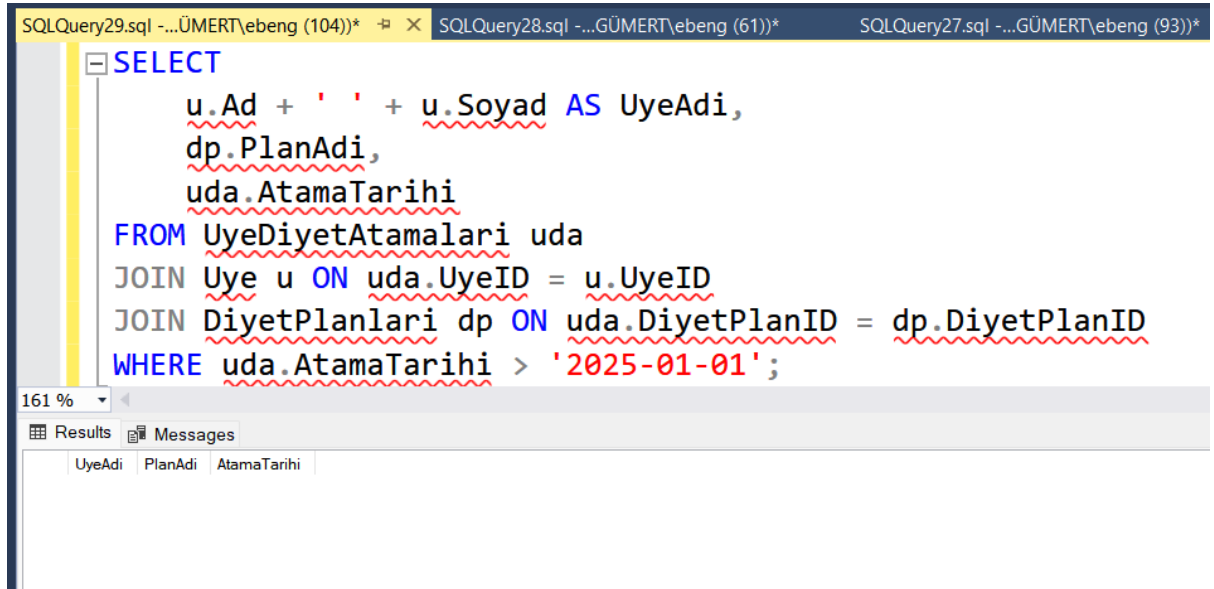
```
SELECT
    dp.PlanAdi,
    dp.GunlukKalori,
    uda.AtamaTarihi
FROM UyeDiyetAtamalari uda
JOIN DiyetPlanlari dp ON uda.DiyetPlanID = dp.DiyetPlanID
WHERE uda.UyeID = 3;
```

The results pane shows the following data:

	PlanAdi	GunlukKalori	AtamaTarihi
1	Kas Yapma Planı	2500	2024-03-05
2	Kilo Verme Planı	1500	2024-08-01
3	Kilo Verme Planı	1500	2024-08-01

**Şekil-29:Üye numarası 3 olan üyeye atanan diyet listelerini gösteren SQL komutu**

Bu sorgu, belirli bir tarihten (örneğin '2025-01-01') sonra yapılan tüm diyet planı atamalarını listelemektedir. Böylece son dönemde yapılan atamalar incelenebilmektedir.



The screenshot shows a SQL query window with the following query:

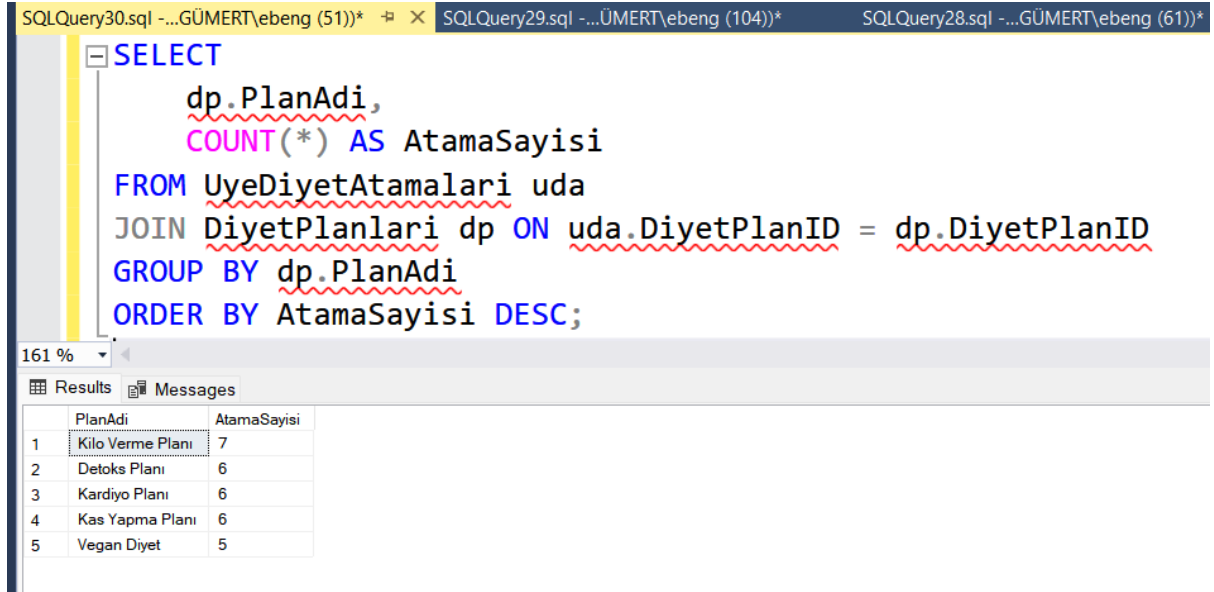
```
SELECT
    u.Ad + ' ' + u.Soyad AS UyeAdi,
    dp.PlanAdi,
    uda.AtamaTarihi
FROM UyeDiyetAtamalari uda
JOIN Uye u ON uda.UyeID = u.UyeID
JOIN DiyetPlanlari dp ON uda.DiyetPlanID = dp.DiyetPlanID
WHERE uda.AtamaTarihi > '2025-01-01';
```

The results pane shows the following data:

UyeAdi	PlanAdi	AtamaTarihi
--------	---------	-------------

**Şekil-30:Belli bir tarihten sonra yapılan tüm diyet planı atamalarını gösteren SQL komutu**

Aşağıdaki SQL sorgusunda, her bir diyet planının kaç farklı üyeye atandığı hesaplanmıştır. Bu sorgu sayesinde hangi diyet planının daha yoğun kullanıldığı analiz edilebilmektedir.



```
SQLQuery30.sql - ...GÜMERT\ebeng (51))* SQLQuery29.sql - ...ÜMERT\ebeng (104))* SQLQuery28.sql - ...GÜMERT\ebeng (61))*  
SELECT  
    dp.PlanAdi,  
    COUNT(*) AS AtamaSayisi  
FROM UyeDiyetAtamaları uda  
JOIN DiyetPlanları dp ON uda.DiyetPlanID = dp.DiyetPlanID  
GROUP BY dp.PlanAdi  
ORDER BY AtamaSayisi DESC;
```

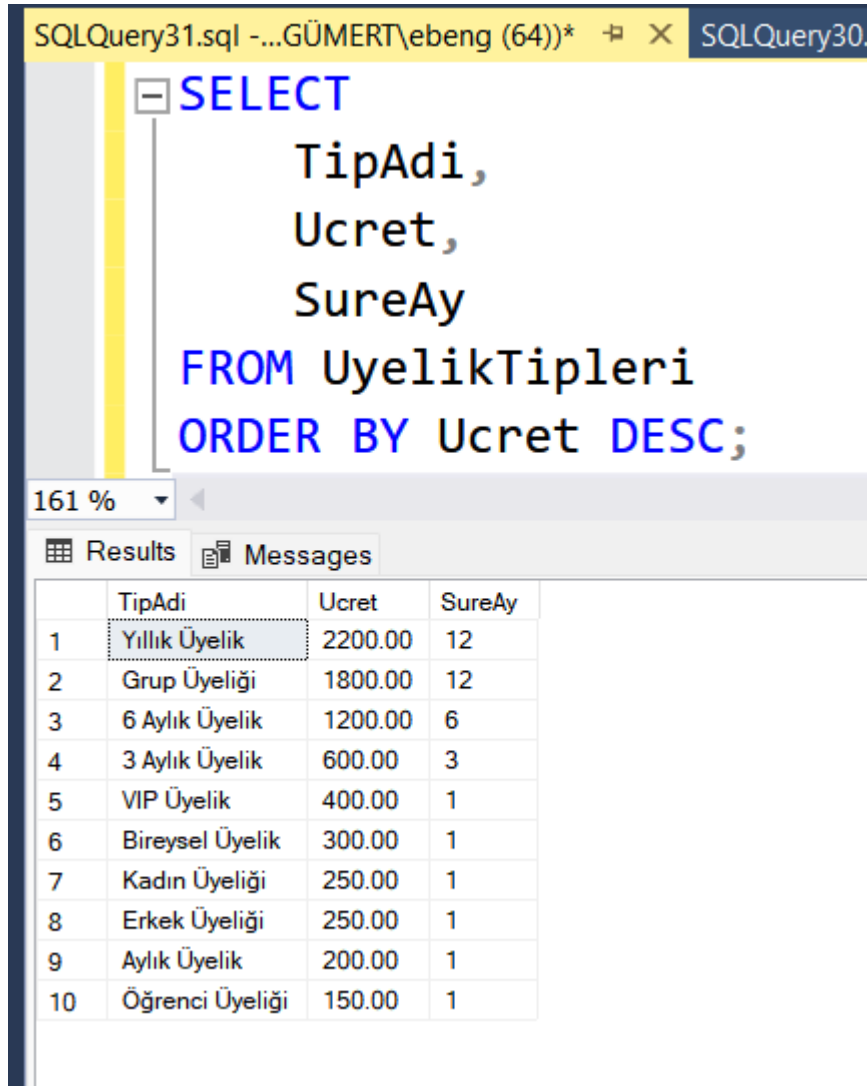
161 %

Results Messages

	PlanAdi	AtamaSayisi
1	Kilo Verme Planı	7
2	Detoks Planı	6
3	Kardiyo Planı	6
4	Kas Yapma Planı	6
5	Vegan Diyet	5

**Şekil-31: Her bir diyet planının kaç farklı üyeye atandığını gösteren SQL komutu**

Aşağıdaki sorgu, UyelikTipleri tablosundan en yüksek ücretli üyelikleri sıralayarak listelemektedir. Böylece en pahalı üyelik türleri analiz edilebilir.



The screenshot shows a SQL query window with the following text:

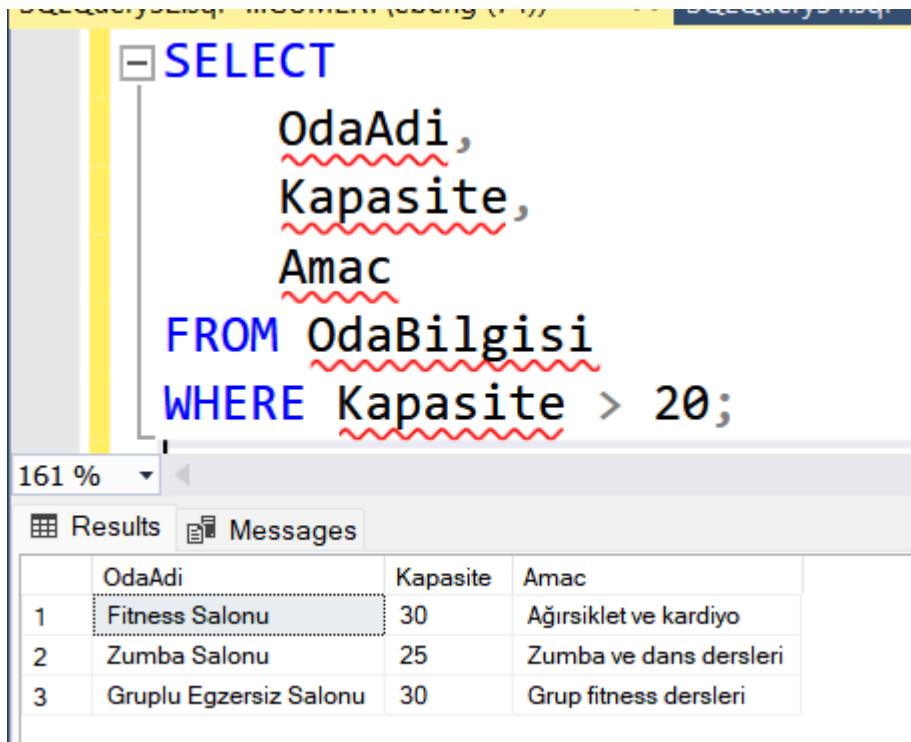
```
SELECT
    TipAdi,
    Ucret,
    SureAy
FROM UyelikTipleri
ORDER BY Ucret DESC;
```

Below the query, the 'Results' tab is active, displaying a table with 10 rows. The columns are TipAdi, Ucret, and SureAy. The rows are sorted by Ucret in descending order.

	TipAdi	Ucret	SureAy
1	Yıllık Üyelik	2200.00	12
2	Grup Üyeliği	1800.00	12
3	6 Aylık Üyelik	1200.00	6
4	3 Aylık Üyelik	600.00	3
5	VIP Üyelik	400.00	1
6	Bireysel Üyelik	300.00	1
7	Kadın Üyeliği	250.00	1
8	Erkek Üyeliği	250.00	1
9	Aylık Üyelik	200.00	1
10	Öğrenci Üyeliği	150.00	1

Şekil-32: En yüksek ücretli üyelikleri sıralayarak listeleyen SQL komutu

Aşağıdaki sorgu, OdaBilgisi tablosundan kapasitesi 20'den büyük olan odaların adlarını ve kullanım amaçlarını listelemektedir.



```
SELECT
    OdaAdi,
    Kapasite,
    Amac
FROM OdaBilgisi
WHERE Kapasite > 20;
```

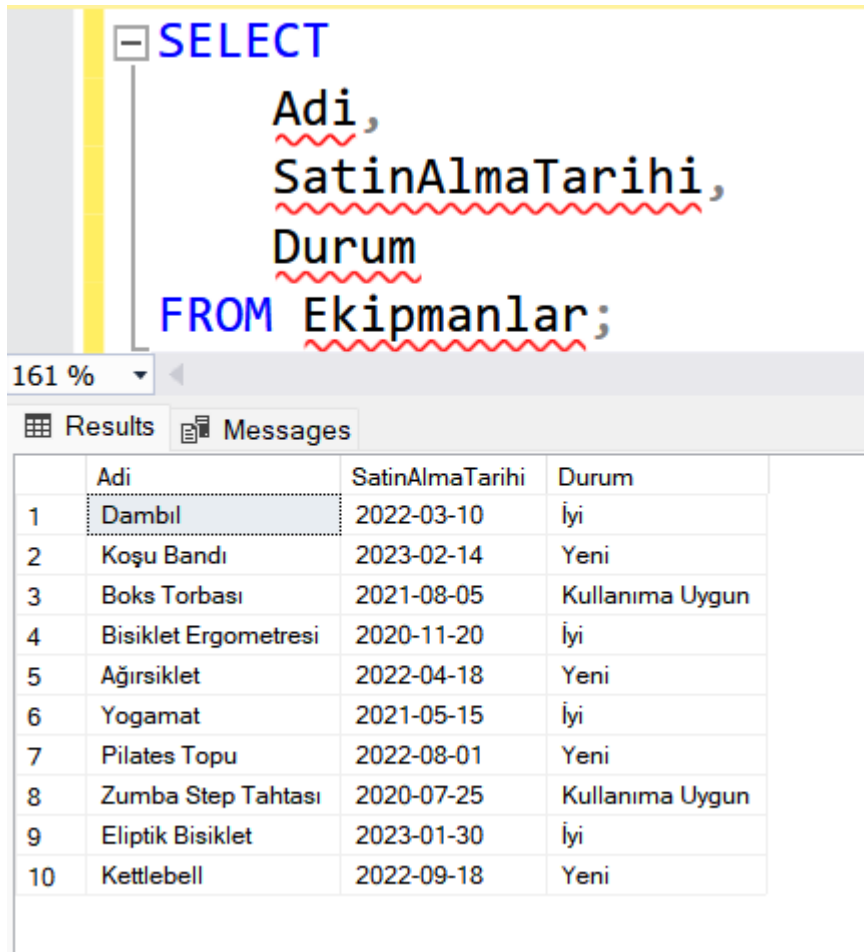
161 %

Results Messages

	OdaAdi	Kapasite	Amac
1	Fitness Salonu	30	Ağırsiklet ve kardiyo
2	Zumba Salonu	25	Zumba ve dans dersleri
3	Gruplu Egzersiz Salonu	30	Grup fitness dersleri

Şekil-33: Kapasitesi 20'den fazla olan salonları listeleyen SQL komutu

Aşağıdaki sorgu, Ekipmanlar tablosundaki ekipmanların adları, satın alma tarihleri ve mevcut durumlarını göstermektedir. Bu sayede ekipman takibi sağlanabilir.



The screenshot shows a SQL query editor with a query window and a results pane. The query is:

```
SELECT  
    Adi,  
    SatinAlmaTarihi,  
    Durum  
FROM Ekipmanlar;
```

The results pane shows a table with 10 rows and 4 columns: Adi, SatinAlmaTarihi, and Durum. The first row is highlighted.

	Adi	SatinAlmaTarihi	Durum
1	Dambıl	2022-03-10	İyi
2	Koşu Bandı	2023-02-14	Yeni
3	Boks Torbası	2021-08-05	Kullanıma Uygun
4	Bisiklet Ergometresi	2020-11-20	İyi
5	Ağırsiklet	2022-04-18	Yeni
6	Yogamat	2021-05-15	İyi
7	Pilates Topu	2022-08-01	Yeni
8	Zumba Step Tahtası	2020-07-25	Kullanıma Uygun
9	Eliptik Bisiklet	2023-01-30	İyi
10	Kettlebell	2022-09-18	Yeni

**Şekil-34: Mevcut ekipmanların durumunu listeleme SQL komutu**

Aşağıdaki sorgu, Takip tablosundan son 7 gün içerisindeki giriş ve çıkış kayıtlarını listelemektedir. Böylece üyelerin son hafta içindeki salon kullanımı gözlemlenebilir.

```
SELECT
    u.Ad + ' ' + u.Soyad AS UyeAdi,
    t.Giris,
    t.Cikis
FROM Takip t
JOIN Uye u ON t.UyeID = u.UyeID
WHERE t.Giris >= DATEADD(DAY, -7, GETDATE());
```

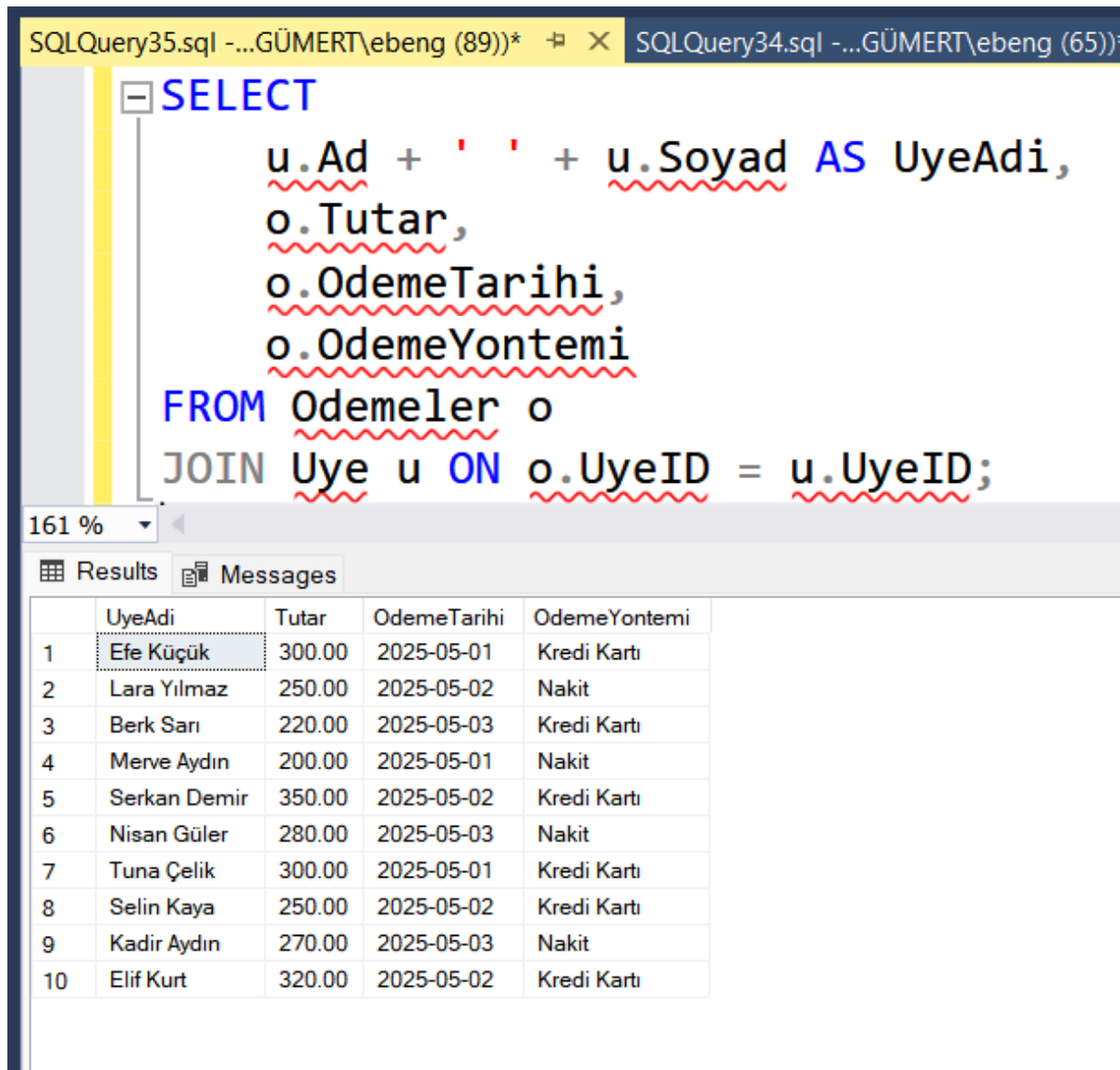
161 %

Results Messages

	UyeAdi	Giris	Cikis
1	Efe Küçük	2025-05-01 09:00:00.000	2025-05-01 10:00:00.000
2	Lara Yılmaz	2025-05-02 08:30:00.000	2025-05-02 09:30:00.000
3	Berk Sarı	2025-05-01 10:00:00.000	2025-05-01 11:00:00.000
4	Merve Aydın	2025-05-02 09:30:00.000	2025-05-02 10:30:00.000
5	Serkan Demir	2025-05-03 10:15:00.000	2025-05-03 11:15:00.000
6	Nisan Güler	2025-05-03 08:00:00.000	2025-05-03 09:00:00.000
7	Tuna Çelik	2025-05-01 11:00:00.000	2025-05-01 12:00:00.000
8	Selin Kaya	2025-05-01 14:00:00.000	2025-05-01 15:00:00.000
9	Kadir Aydın	2025-05-02 07:30:00.000	2025-05-02 08:30:00.000
10	Elif Kurt	2025-05-02 16:00:00.000	2025-05-02 17:00:00.000

Şekil-35: Günlük giriş-çıkış takibi (Son 7 Gün) SQL komutu

Aşağıdaki sorgu, Ödemeler tablosu ile Üye tablosu birleştirilerek oluşturulmuştur. Hangi üyenin, ne kadar ve hangi tarihte ödeme yaptığı bilgisi elde edilmektedir.



The screenshot displays the SQL Server Enterprise Manager interface. At the top, two tabs are visible: 'SQLQuery35.sql - ...GÜMERT\ebeng (89))' and 'SQLQuery34.sql - ...GÜMERT\ebeng (65))'. The main area shows an SQL query in a text editor. The query is as follows:

```
SELECT
    u.Ad + ' ' + u.Soyad AS UyeAdi,
    o.Tutar,
    o.OdemeTarihi,
    o.OdemeYontemi
FROM Ödemeler o
JOIN Üye u ON o.ÜyeID = u.ÜyeID;
```

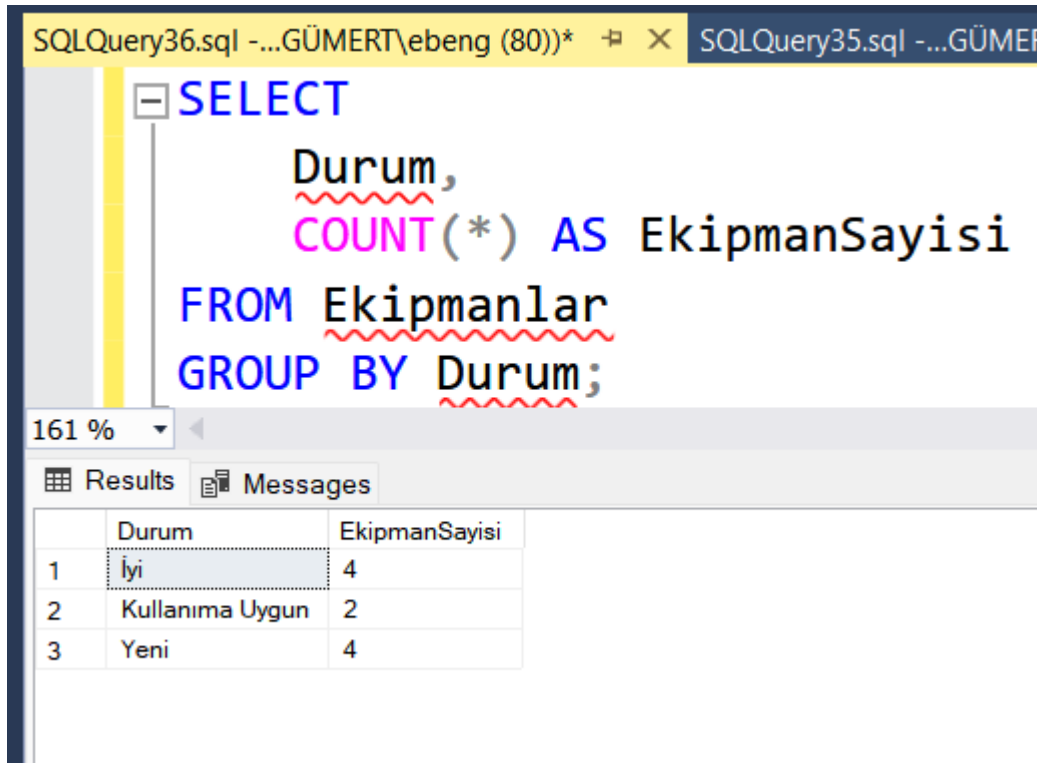
Below the query editor, the 'Results' tab is selected, showing a table with 10 rows of data. The table has five columns: 'UyeAdi', 'Tutar', 'OdemeTarihi', and 'OdemeYontemi'. The data is as follows:

	UyeAdi	Tutar	OdemeTarihi	OdemeYontemi
1	Efe Küçük	300.00	2025-05-01	Kredi Kartı
2	Lara Yılmaz	250.00	2025-05-02	Nakit
3	Berk Sarı	220.00	2025-05-03	Kredi Kartı
4	Merve Aydın	200.00	2025-05-01	Nakit
5	Serkan Demir	350.00	2025-05-02	Kredi Kartı
6	Nisan Güler	280.00	2025-05-03	Nakit
7	Tuna Çelik	300.00	2025-05-01	Kredi Kartı
8	Selin Kaya	250.00	2025-05-02	Kredi Kartı
9	Kadir Aydın	270.00	2025-05-03	Nakit
10	Elif Kurt	320.00	2025-05-02	Kredi Kartı

Şekil-36: Ödeme bilgilerini listeleyen (Üye Adıyla Birlikte) SQL komutu



Aşağıdaki sorgu ile Ekipmanlar tablosunda yer alan ekipmanlar, **durum bilgisine göre gruplanmakta** ve her durumdan **kaç adet olduğu** belirlenmektedir. Bu sayede bozuk, çalışır ya da bakımda olan ekipmanların sayısal dağılımı kolaylıkla takip edilebilir.



The screenshot shows a SQL query window with the following text:

```
SELECT
    Durum,
    COUNT(*) AS EkipmanSayisi
FROM Ekipmanlar
GROUP BY Durum;
```

Below the query, the 'Results' tab is active, displaying a table with 3 rows and 2 columns: 'Durum' and 'EkipmanSayisi'.

	Durum	EkipmanSayisi
1	İyi	4
2	Kullanıma Uygun	2
3	Yeni	4

**Şekil-37: Ekipman durumlarına göre sayısal dağılımı görüntüleme SQL komutu**

Aşağıdaki sorgu ile Personel tablosunda yer alan çalışanlar, işe başladıkları yıl bazında gruplanmakta ve her yılda kaç kişinin işe başladığı ile o yıldan bugüne ortalama kaç yıl çalıştıkları hesaplanmaktadır. Bu sorgu ile kurumun işe alım dönemleri ve çalışan sadakati analiz edilebilir.

```

SELECT
    YEAR(IseAlimTarihi) AS IseBaslamaYili,
    COUNT(*) AS PersonelSayisi,
    AVG(DATEDIFF(YEAR, IseAlimTarihi, GETDATE())) AS OrtalamaCalismaYili
FROM Personel
GROUP BY YEAR(IseAlimTarihi);

```

	IseBaslamaYili	PersonelSayisi	OrtalamaCalismaYili
1	2019	2	6
2	2020	4	5
3	2021	6	4
4	2022	4	3
5	2023	4	2

**Şekil-38: Personellerin işe başlama yılına göre kişi sayısı ve ortalama çalışma süresini hesaplayan SQL komutu**

Aşağıdaki UPDATE SQL komutu ile zam işlemi veritabanı üzerinde **kalıcı olarak uygulanmakta**, yani UyelikTipleri tablosunda yer alan her bir üyelik türünün Ücret değeri belirli bir oran üzerinden güncellenmektedir.

```

UPDATE UyelikTipleri
SET Ücret = Ücret * 1.15;

```

(10 rows affected)

Completion time: 2025-05-03T21:43:33.8097098+03:00

**Şekil-39: Belirli bir zam oranı üzerinden üyelik tiplerinin ücretine zam yapan SQL komutu**

SQLQuery39.sql -...GÜMERT\ebeng (82))\* X SQLQuery38.sql -...GÜMERT\ebeng (60))\*

**SELECT \* FROM UyelikTipleri;**

161 %

Results Messages

	UyelikTipiID	TipAdi	Ucret	SureAy
1	1	Aylık Üyelik	230.00	1
2	2	Yıllık Üyelik	2530.00	12
3	3	3 Aylık Üyelik	690.00	3
4	4	6 Aylık Üyelik	1380.00	6
5	5	Öğrenci Üyeliği	172.50	1
6	6	VIP Üyelik	460.00	1
7	7	Grup Üyeliği	2070.00	12
8	8	Bireysel Üyelik	345.00	1
9	9	Kadın Üyeliği	287.50	1
10	10	Erkek Üyeliği	287.50	1

**Şekil-40: Belirli bir zam oranı üzerinden üyelik tiplerinin ücretine zam yapan SQL komutu çalıştırıldıktan sonra**

## 8.Saklı Yordamlar( Stored procedures) Örnekleri

Aşağıdaki saklı yordam, belirli bir üye için ödeme bilgilerini Odemeler tablosuna ekler.

SQLQuery40.sql -...GÜMERT\ebeng (66))\* X SQLQuery39.sql -...GÜMERT\ebeng (82))\* SQLQuery38.sql -...GÜMERT\ebeng (60))\* SQLQuery37

```

CREATE PROCEDURE OdemeEkle
    @UyeID INT,
    @Tutar DECIMAL(10,2),
    @OdemeTarihi DATE,
    @OdemeYontemi VARCHAR(50)
AS
BEGIN
    INSERT INTO Odemeler (UyeID, Tutar, OdemeTarihi, OdemeYontemi)
    VALUES (@UyeID, @Tutar, @OdemeTarihi, @OdemeYontemi);
END;

```

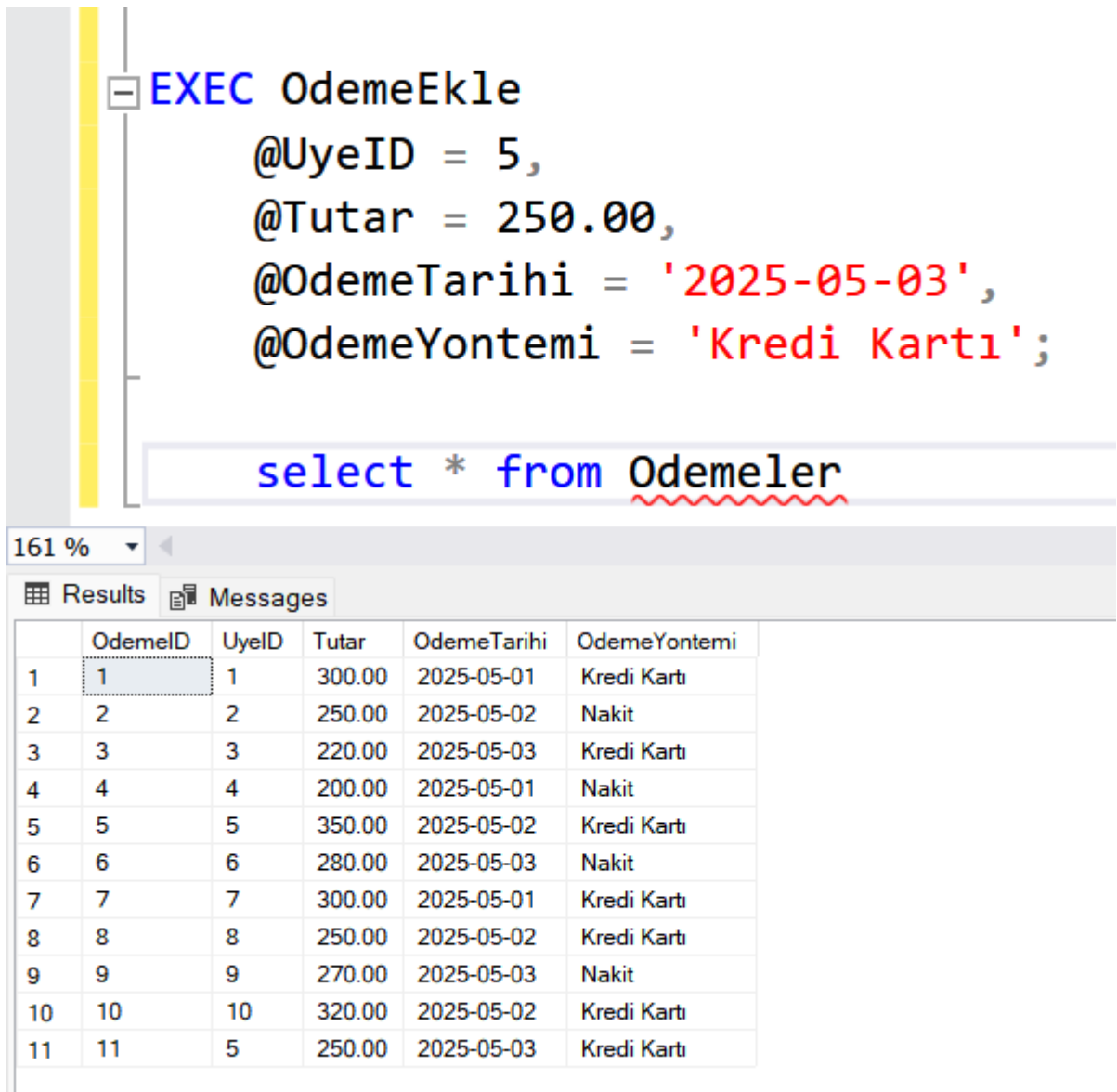
161 %

Messages

Commands completed successfully.

Completion time: 2025-05-03T21:54:30.1883243+03:00

Şekil-41: Üye Ödeme Bilgisi Ekleme Saklı Yordamı



The screenshot shows a SQL query editor with a stored procedure call and its results. The query is as follows:

```
EXEC OdemeEkle
    @UyeID = 5,
    @Tutar = 250.00,
    @OdemeTarihi = '2025-05-03',
    @OdemeYontemi = 'Kredi Kartı';

select * from Odemeler
```

The results are displayed in a table with the following columns: OdemeID, UyeID, Tutar, OdemeTarihi, and OdemeYontemi. The table contains 11 rows of data.

	OdemeID	UyeID	Tutar	OdemeTarihi	OdemeYontemi
1	1	1	300.00	2025-05-01	Kredi Kartı
2	2	2	250.00	2025-05-02	Nakit
3	3	3	220.00	2025-05-03	Kredi Kartı
4	4	4	200.00	2025-05-01	Nakit
5	5	5	350.00	2025-05-02	Kredi Kartı
6	6	6	280.00	2025-05-03	Nakit
7	7	7	300.00	2025-05-01	Kredi Kartı
8	8	8	250.00	2025-05-02	Kredi Kartı
9	9	9	270.00	2025-05-03	Nakit
10	10	10	320.00	2025-05-02	Kredi Kartı
11	11	5	250.00	2025-05-03	Kredi Kartı

Şekil-42: Üye Ödeme Bilgisi Ekleme Saklı Yordamı Çalıştırdıktan Sonra  
'Odemeler'tablosu

Aşağıdaki örnekte, **Uye** tablosuna yeni bir üye eklemek için bir saklı yordam oluşturulmuştur. Yordam, parametre olarak alınan üyelik bilgilerini tablodaki ilgili sütunlara ekler.

```
SQLQuery42.sql - ...GÜMERT\ebeng (81))* x SQLQuery41.sql - ...ÜMERT\ebeng (112)) SQLQuery40.sql - ...GÜMERT\ebeng (66))* SQLQuery39.sql - ...GÜMERT\ebeng (82))*  
CREATE PROCEDURE UyeEkle  
    @Ad VARCHAR(50),  
    @Soyad VARCHAR(50),  
    @Cinsiyet VARCHAR(10),  
    @DogumTarihi DATE,  
    @Telefon VARCHAR(15),  
    @Eposta VARCHAR(100),  
    @KayitTarihi DATE,  
    @UyelikTipiID INT  
AS  
BEGIN  
    INSERT INTO Uye (Ad, Soyad, Cinsiyet, DogumTarihi, Telefon, Eposta, KayitTarihi, UyelikTipiID)  
    VALUES (@Ad, @Soyad, @Cinsiyet, @DogumTarihi, @Telefon, @Eposta, @KayitTarihi, @UyelikTipiID);  
END;  
  
161 %  
Messages  
Commands completed successfully.  
Completion time: 2025-05-03T21:59:31.6750021+03:00
```

Şekil-43: Üye Ekleme Saklı Yordamı

```
EXEC UyeEkle  
    @Ad = 'Ahmet',  
    @Soyad = 'Yılmaz',  
    @Cinsiyet = 'E',  
    @DogumTarihi = '1990-05-15',  
    @Telefon = '5551234567',  
    @Eposta = 'ahmet@mail.com',  
    @KayitTarihi = '2025-05-03',  
    @UyelikTipiID = 2;  
  
select * from Uye
```

161 %

Results Messages Client Statistics

	UyeID	Ad	Soyad	Cinsiyet	DogumTarihi	Telefon	Eposta	KayitTarihi	UyelikTipiID
1	1	Efe	Küçük	Erkek	1995-04-12	5558765432	efe@example.com	2025-04-20	1
2	2	Lara	Yılmaz	Kadın	1998-07-20	5559876543	lara@example.com	2025-04-22	2
3	3	Berk	Sarı	Erkek	1994-11-15	5556543210	berk@example.com	2025-04-21	1
4	4	Merve	Aydın	Kadın	2000-01-10	5553219876	merve@example.com	2025-04-23	2
5	5	Serkan	Demir	Erkek	1990-03-22	5555678901	serkan@example.com	2025-04-19	3
6	6	Nisan	Güler	Kadın	1997-05-17	5552348901	nisan@example.com	2025-04-24	1
7	7	Tuna	Çelik	Erkek	1989-12-09	5558765432	tuna@example.com	2025-04-18	2
8	8	Selin	Kaya	Kadın	1993-06-30	5556541234	selin@example.com	2025-04-15	1
9	9	Kadir	Aydın	Erkek	1992-10-05	5554321098	kadir@example.com	2025-04-14	3
10	10	Elif	Kurt	Kadın	1996-04-22	5559876543	elif@example.com	2025-04-17	2
11	11	Ahmet	Yılmaz	E	1990-05-15	5551234567	ahmet@mail.com	2025-05-03	2

Şekil-44: Saklı Yordam Çalıştırıldıktan Sonra 'Uye' tablosu

Aşağıdaki saklı yordam, **OdaBilgisi** tablosundaki odaları kapasiteye göre sıralar.

```
CREATE PROCEDURE OdaKapasitesiListele
AS
BEGIN
    SELECT
        OdaAdi,
        Kapasite
    FROM OdaBilgisi
    ORDER BY Kapasite DESC;
END;
EXEC OdaKapasitesiListele;
```

161 %

Results Messages

	OdaAdi	Kapasite
1	Fitness Salonu	30
2	Gruplu Egzersiz Salonu	30
3	Zumba Salonu	25
4	Yoga Salonu	20
5	Ağırsiklet Salonu	18
6	Boks Salonu	15
7	Pilates Salonu	12
8	Koşu Parkuru	10
9	Dinlenme Salonu	8
10	Sauna ve Buhar Odası	5

**Şekil-44:** OdaBilgisi tablosundaki odaları kapasiteye göre sıralayan saklı yordam

## 8.Tetikleyici (Trigger) Örneği

### Senaryo: Ödeme Yapıldığında Log Tutma

Bir üye ödeme yaptığında, bu işlemi bir log tablosunda otomatik olarak kaydetmek isteyebilirsiniz. Bu amaçla Odemeler tablosuna **AFTER INSERT** tetikleyicisi tanımlayacağız.

#### 1. Log Tablosunun Oluşturulması

Öncelikle log kayıtlarını tutmak için bir tablo oluşturalım:

```
CREATE TABLE OdemeLog (  
    LogID INT PRIMARY KEY IDENTITY,  
    UyeID INT,  
    Tutar DECIMAL(10,2),  
    OdemeTarihi DATE,  
    IslemTarihi DATETIME DEFAULT GETDATE()  
);
```

161 %

Messages

Commands completed successfully.

Completion time: 2025-05-03T22:21:37.0649877+03:00

#### 2. Tetikleyicinin Oluşturulması

Şimdi Odemeler tablosuna bir kayıt eklendiğinde otomatik olarak OdemeLog tablosuna veri ekleyen trigger'ı yazalım:

```
CREATE TRIGGER trg_OdemeEklendigindeLogla
ON Odemeler
AFTER INSERT
AS
BEGIN
    INSERT INTO OdemeLog (UyeID, Tutar, OdemeTarihi)
    SELECT UyeID, Tutar, OdemeTarihi
    FROM INSERTED;
END;
```

161 %

Messages

Commands completed successfully.

Completion time: 2025-05-03T22:22:24.4639190+03:00

### 3. Nasıl Çalışır?

```
INSERT INTO Odemeler (UyeID, Tutar, OdemeTarihi, OdemeYontemi)
VALUES (3, 300.00, '2025-05-03', 'Nakit');
```

```
SELECT * FROM OdemeLog;
```

161 %

Results Messages

	LogID	UyeID	Tutar	OdemeTarihi	IslemTarihi
1	1	3	300.00	2025-05-03	2025-05-03 22:23:20.337

Şekil-45: Trigger çalıştıktan sonra OdemeLog tablosu