T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR ve BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

Veri Tabanı Yönetim Sistemleri Dersi Proje Raporu

ÖDEVİN KONUSU: Kuaför Randevu(Yönetim) Sistemi uygulaması.

ÖĞRENCİNİN ADI SOYADI: Ahmet Tarık Türkmen

NUMARASI: G221210087

SINIFI: 2.Öğretim C Grubu

DERSİ VEREN: Prof.Dr. CELAL ÇEKEN

E-POSTA: tarik.turkmen@ogr.sakarya.edu.tr

Github: https://github.com/ahmettarikturkmen/Veri-Taban--Proje-Kuafor-db

SAKARYA, 2024

Senaryo

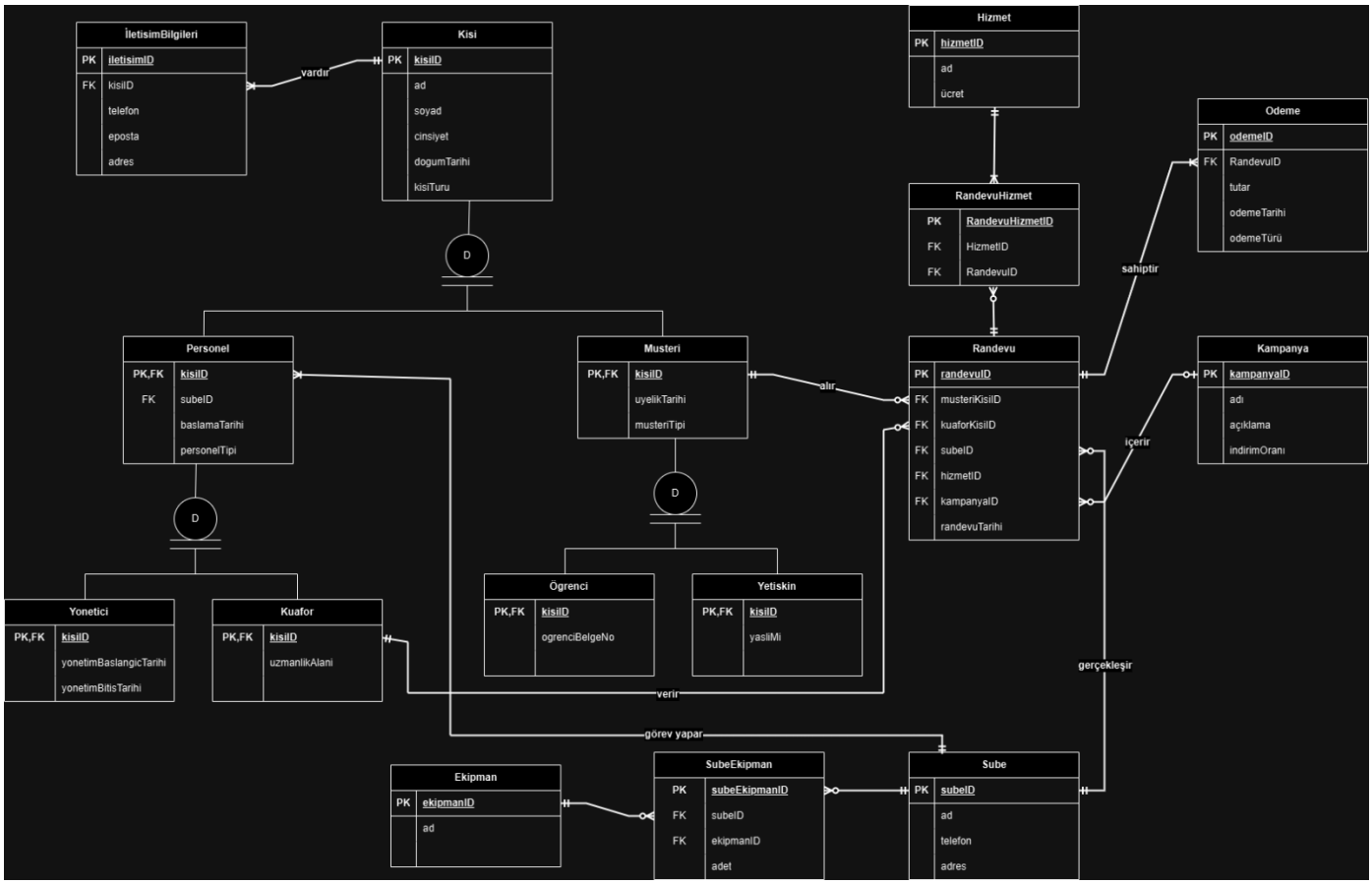
Bir Kuaför dükkanı sahibi dükkanındaki çalışanların, gelen müşterilerin ve hizmetlerin bilgilerini saklamak, yönetmek için bir yazılım sistemi istemektedir.

İş kuralları

- Bu veri tabanında her bir kişi ya müşteri ya da personeldir. Bir kişi hem personel hem müşteri olamaz. Bu kişilerin adı, soyadı, cinsiyeti, doğum tarihi saklanacaktır.
- Bir personel hem yönetici hem kuaför olamaz. Yönetici ve kuaför olmayan personel olamaz. Yöneticilerin başlangıç tarihi ve bitiş tarihi saklanacaktır. Kuaförlerin ise uzmanlık alanı kaydedilecektir.
- Öğrenci veya yetişkin olmayan müşteri olamaz. Bir müşteri hem öğrenci hem yetişkin olamaz. Öğrencilerin öğrenci belgesi numarası kaydedilecektir. Yetişkinlerin de 60 yaş üstünde olup olmadıkları kaydedilecektir.
- Kişilerin e-mail, telefon ve adres bilgileri mevcuttur.
- Müşteriler hizmeti kuaförlerden almak için randevu almaları gerekir. Her randevunun eşsiz bir randevu numarası, randevu tarihi ve saati mevcuttur.
- Personeller tarafından verilen hizmetlerin(saç tıraşı, sakal tıraşı, ağda vb.), kodu, adı, ücreti mevcuttur. Hizmetler belli şartlarda kampanyaya dahil olurlar.
- Kampanyalar iki tanedir. İlki Öğrenci indirimi: Tüm öğrencilere özel %20 indirim, ikincisi yaşlı indirimi: eğer bir yetişkin yaşlıysa yani 60 yaşından büyükse ona yüzde 20 indirim uygulanacak. Her müşteri en fazla bir indirimden faydalanabilir.
- Bu işletmenin bir çok şubesi vardır. Her kuaför belirli bir şubede çalışır. Bir şubede birden fazla kuaför çalışabilir.
- Kullanılan ekipman türlerinin tutulması gerekmektedir. Ekipmanların adı, adedi, bulunduğu şube kaydedilecektir. Bir şube birden çok ekipman türüne sahip olabilir. Aynı ekipman birden fazla şubede bulunabilir.
- Hizmet sonucu müşterinin ödeme bilgileri; toplam tutar, ödeme türü, ödeme tarihi kaydedilecektir. Taksitli ödeme yapılabilir.

İlişkisel Şema

- Kisi(kisiID:int,ad:varchar,soyad:varchar,cinsiyet:varchar,dogumTarihi:date,kisiTuru:enum)
- İletisimBilgileri(iletisimID:int,kisiID:int,telefon:varchar,eposta:varchar,adres:varchar)
- Personel(kisiID:int,subeID:int,baslamaTarihi:date,personelTipi:enum)
- Musteri(kisiID:int,uyelikTarihi:date,musteriTipi:enum)
- Yonetici(kisiID:int,yonetimBaslamaTarihi:date,yonetimBitisTarihi:date)
- Kuafor(kisiID:int,uzmanlikAlani:varchar)
- Ogrenci(kisiID:int,ogrenciBelgeNo:varchar)
- Yetiskin(kisiID:int,yasliMi:boolean)
- Randevu(randevuID:int,musteriKisiID:int,berberKisiID:int,subeID:int,kampanyaID:int,hizmetID:int,randevuTarihi:timestamp)
- Hizmet(hizmetID:int,ad:varchar,ücret:money)
- RandevuHizmet(randevuHizmetID:int,hizmetID:int,randevuID:int)
- Odeme(odemeID:int,randevuID:int,tutar:money,odemeTarihi:date,odemeTuru:varchar)
- Kampanya(kampanyaID:int,adı:varchar,acıklama:varchar,indirimOranı:numeric(5.2))
- Sube(subeID:int,ad:varchar,telefon:varchar,adres:varchar)
- Ekipman(ekipmanID:int,ad:varchar)
- SubeEkipman(subeEkipmanID:int,subeID:int,ekipmanID:int,adet:int)



Veritabanını, içerisindeki verilerle birlikte oluşturmayı sağlayan SQL ifadeleri

CREATE DATABASE "KuaforSistemi2"

WITH

OWNER = postgres

ENCODING = 'UTF8'

LC_COLLATE = 'Turkish_Türkiye.1254'

LC_CTYPE = 'Turkish_Türkiye.1254'

LOCALE_PROVIDER = 'libc'

TABLESPACE = pg_default

CONNECTION LIMIT = -1

IS_TEMPLATE = False;

```
CREATE TABLE Kisi (
    KisiID SERIAL PRIMARY KEY,
    Ad VARCHAR(50) NOT NULL,
    Soyad VARCHAR(50) NOT NULL,
    Cinsiyet VARCHAR(10),
    DogumTarihi DATE NOT NULL,
    KisiTuru VARCHAR(10) NOT NULL
);
CREATE TABLE IletisimBilgileri (
    IletisimID SERIAL PRIMARY KEY,
    KisiID INT NOT NULL,
    Telefon VARCHAR(15),
    Eposta VARCHAR(100),
    Adres TEXT,
    FOREIGN KEY (KisiID) REFERENCES Kisi(KisiID) ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Personel (
    KisiID INT PRIMARY KEY,
    SubeID INT NOT NULL,
    BaslamaTarihi DATE NOT NULL,
    PersonelTipi VARCHAR(10) NOT NULL,
    FOREIGN KEY (KisiID) REFERENCES Kisi(KisiID) ON DELETE CASCADE ON UPDATE CASCADE,--Eğer KisiID'si Personel tablosunda güncellenirse, Yonetici ve Kuafor tablolarındaki KisiID de otomatik olarak güncellenir.
    FOREIGN KEY (SubeID) REFERENCES Sube(SubeID)
);


CREATE TABLE Sube (
```

```
    SubeID SERIAL PRIMARY KEY,
    Ad VARCHAR(100) NOT NULL,
    Telefon VARCHAR(15),
    Adres TEXT
);
CREATE TABLE Yonetici (
    KisiID INT PRIMARY KEY,
    YonetimBaslamaTarihi DATE NOT NULL,
    YonetimBitisTarihi DATE,
    FOREIGN KEY (KisiID) REFERENCES Personel(KisiID) ON DELETE CASCADE ON
UPDATE CASCADE
);
CREATE TABLE Kuafor (
    KisiID INT PRIMARY KEY,
    UzmanlikAlani VARCHAR(100),
    FOREIGN KEY (KisiID) REFERENCES Personel(KisiID) ON DELETE CASCADE ON
UPDATE CASCADE
);
CREATE TABLE Musteri (
    KisiID INT PRIMARY KEY,
    UyelikTarihi DATE NOT NULL,
    MusteriTipi VARCHAR(10) NOT NULL,
    FOREIGN KEY (KisiID) REFERENCES Kisi(KisiID) ON DELETE CASCADE ON UPDATE
CASCADE
);
CREATE TABLE Ogrenci (
    KisiID INT PRIMARY KEY,
    OgrenciBelgeNo VARCHAR(50) NOT NULL,
    FOREIGN KEY (KisiID) REFERENCES Musteri(KisiID) ON DELETE CASCADE ON
UPDATE CASCADE
```

```
);
CREATE TABLE Yetiskin (
    KisiID INT PRIMARY KEY,
    YasliMi BOOLEAN NOT NULL, -- 60 yaş üzeri 1, aksi halde 0
    FOREIGN KEY (KisiID) REFERENCES Musteri(KisiID) ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Ekipman (
    EkipmanID SERIAL PRIMARY KEY,
    Ad VARCHAR(100) NOT NULL
);
CREATE TABLE SubeEkipman (
    SubeEkipmanID SERIAL PRIMARY KEY,  -- Yeni birincil anahtar ekleniyor
    SubeID INT NOT NULL,               -- Subesinin ID'si
    EkipmanID INT NOT NULL,            -- Ekipmanın ID'si
    Adet INT NOT NULL,                 -- Adet bilgisi
    CONSTRAINT unique_sube_ekipman UNIQUE (SubeID, EkipmanID),  -- SubeID ve EkipmanID kombinasyonu UNIQUE olacak
    FOREIGN KEY (SubeID) REFERENCES Sube(SubeID) ON DELETE CASCADE ON UPDATE CASCADE,
    FOREIGN KEY (EkipmanID) REFERENCES Ekipman(EkipmanID) ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE Hizmet (
    HizmetID SERIAL PRIMARY KEY,
    Ad VARCHAR(100) NOT NULL,
    Ucret MONEY NOT NULL
);


CREATE TABLE Kampanya (
```

```
    KampanyaID SERIAL PRIMARY KEY,
    Ad VARCHAR(100) NOT NULL,
    Aciklama TEXT,
    IndirimOrani NUMERIC(5, 2) NOT NULL
);
CREATE TABLE Randevu (
    RandevuID SERIAL PRIMARY KEY,
    MusteriKisiID INT NOT NULL,
    BerberKisiID INT NOT NULL,
    SubeID INT NOT NULL,
    KampanyaID INT,
    RandevuTarihi TIMESTAMP NOT NULL,
    FOREIGN KEY (MusteriKisiID) REFERENCES Musteri(KisiID) ON DELETE CASCADE
ON UPDATE CASCADE,
    FOREIGN KEY (BerberKisiID) REFERENCES Personel(KisiID) ON DELETE CASCADE
ON UPDATE CASCADE,
    FOREIGN KEY (SubeID) REFERENCES Sube(SubeID) ON DELETE CASCADE ON
UPDATE CASCADE,
    FOREIGN KEY (KampanyaID) REFERENCES Kampanya(KampanyaID) ON DELETE
CASCADE ON UPDATE CASCADE
);
ALTER TABLE Randevu
ADD COLUMN HizmetID INT;
ALTER TABLE Randevu
ADD CONSTRAINT fk_hizmetID
FOREIGN KEY (HizmetID) REFERENCES Hizmet(hizmetID)
ON DELETE SET NULL ON UPDATE CASCADE;

CREATE TABLE RandevuHizmet (
    RandevuHizmetID SERIAL PRIMARY KEY,   -- Yeni birincil anahtar
```

```
    RandevuID INT NOT NULL,              -- RandevuID foreign key
    HizmetID INT NOT NULL,               -- HizmetID foreign key
    CONSTRAINT fk_randevu FOREIGN KEY (RandevuID) REFERENCES
Randevu(RandevuID) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_hizmet FOREIGN KEY (HizmetID) REFERENCES Hizmet(HizmetID)
ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT uq_randevu_hizmet UNIQUE (RandevuID, HizmetID)   -- RandevuID ve
HizmetID kombinasyonu unique olacak
);

CREATE TABLE Odeme (
    OdemeID SERIAL PRIMARY KEY,
    RandevuID INT NOT NULL,
    Tutar MONEY NOT NULL,
    OdemeTarihi DATE NOT NULL,
    OdemeTuru VARCHAR(20) NOT NULL,
    FOREIGN KEY (RandevuID) REFERENCES Randevu(RandevuID) ON DELETE CASCADE
ON UPDATE CASCADE
);
ALTER TABLE IletisimBilgileri
ADD CONSTRAINT unique_eposta UNIQUE (Eposta);
```

```
INSERT INTO Kisi (Ad, Soyad, Cinsiyet, DogumTarihi, KisiTuru)
VALUES ('Ahmet', 'Yılmaz', 'Erkek', '1990-05-12', 'M');  -- 'M' Müşteri

INSERT INTO Kisi (Ad, Soyad, Cinsiyet, DogumTarihi, KisiTuru)
VALUES ('Zeynep', 'Kaya', 'Kadın', '1985-07-23', 'P');  -- 'P' Personel

INSERT INTO IletisimBilgileri (KisiID, Telefon, Eposta, Adres)
VALUES (1, '0224 123 45 67', 'ahmet@example.com', 'Osmangazi Mah., Bursa');
```

```
INSERT INTO IletisimBilgileri (KisiID, Telefon, Eposta, Adres)
VALUES (2, '0224 234 56 78', 'zeynep@example.com', 'Nilüfer Mah., Bursa');

INSERT INTO Sube (Ad, Telefon, Adres)
VALUES ('Osmangazi Şubesi', '0224 123 45 67', 'Osmangazi Mah., Bursa');

INSERT INTO Sube (Ad, Telefon, Adres)
VALUES ('Nilüfer Şubesi', '0224 234 56 78', 'Nilüfer Mah., Bursa');

INSERT INTO Personel (KisiID, SubeID, BaslamaTarihi, PersonelTipi)
VALUES (2, 1, '2024-01-01', 'K');  -- 'K' Kuaför

INSERT INTO Kuafor (KisiID, UzmanlikAlani)
VALUES (2, 'Saç Kesimi');

INSERT INTO Musteri (KisiID, UyelikTarihi, MusteriTipi)
VALUES (1, '2024-01-01', 'Ogrenci'); -- Öğrenci tipi

INSERT INTO Ogrenci (KisiID, OgrenciBelgeNo)
VALUES (1, 'OG12345');

INSERT INTO Hizmet (Ad, Ucret)
VALUES ('Saç Tıraşı', 100.00);

INSERT INTO Hizmet (Ad, Ucret)
VALUES ('Sakal Tıraşı', 50.00);
```

```
INSERT INTO Randevu (MusteriKisiID, BerberKisiID, SubeID, KampanyaID,
RandevuTarihi)
VALUES (1, 2, 1, NULL, '2024-01-01 10:00:00');

INSERT INTO Odeme (RandevuID, Tutar, OdemeTarihi, OdemeTuru)
VALUES (1, 100.00, '2024-01-01', 'Kredi Kartı');

INSERT INTO Kisi (Ad, Soyad, Cinsiyet, DogumTarihi, KisiTuru)
VALUES ('Ali', 'Veli', 'Erkek', '1955-01-01', 'M');

INSERT INTO Musteri (KisiID, UyelikTarihi, MusteriTipi)
VALUES (
   (SELECT KisiID FROM Kisi WHERE Ad = 'Ali' AND Soyad = 'Veli'),
   CURRENT_DATE,  -- Üyelik tarihi bugün
   'Yetiskin'     -- Müşteri tipi: Yetişkin
);
INSERT INTO Yetiskin (KisiID)
VALUES (
   (SELECT KisiID FROM Kisi WHERE Ad = 'Ali' AND Soyad = 'Veli')
);

INSERT INTO Kisi (Ad, Soyad, Cinsiyet, DogumTarihi, KisiTuru)
VALUES ('Tarik', 'Öğrenci', 'Erkek', '2005-04-12', 'M');

INSERT INTO Musteri (KisiID, UyelikTarihi, MusteriTipi)
VALUES (7, CURRENT_DATE, 'Ogrenci');

INSERT INTO Ogrenci (KisiID, OgrenciBelgeNo)
VALUES (7, 'OG78906');
```

```
ALTER TABLE Odeme
ALTER COLUMN Tutar DROP NOT NULL;

ALTER TABLE Yetiskin
ALTER COLUMN YasliMi DROP NOT NULL;

INSERT INTO Kisi (Ad, Soyad, Cinsiyet, DogumTarihi, KisiTuru)
VALUES ('Emirhan', 'Yucel', 'Erkek', '1950-04-12', 'Musteri');

INSERT INTO Musteri (KisiID, UyelikTarihi, MusteriTipi)
VALUES (9, CURRENT_DATE, 'Yetiskin');

INSERT INTO Yetiskin (KisiID, yasliMi)
VALUES (9,NULL );
INSERT INTO Kuafor (KisiID, UzmanlikAlani)
VALUES (2, 'Saç Kesimi');

INSERT INTO Randevu (MusteriKisiID, BerberKisiID, SubeID, KampanyaID,
RandevuTarihi)
VALUES (9, 2, 1, NULL, '2024-09-05 10:00:00');
INSERT INTO RandevuHizmet (HizmetID, RandevuID)
VALUES (1, 4);

INSERT INTO Randevu (musteriKisiID, berberKisiID, subeID, kampanyaID,
randevuTarihi, hizmetID)
VALUES (3, 2, 2, NULL, '2024-12-21 10:00:00', 2);

INSERT INTO Kampanya ( ad, aciklama, indirimOrani)
```

```
VALUES ('Öğrenci İndirimi', 'Tüm öğrencilere özel %20 indirim.', 20.00);


INSERT INTO Kampanya ( ad, aciklama, indirimOrani)
VALUES ('Yaşlı İndirimi', '60 yaşından büyük yetişkinlere özel %20 indirim.', 20.00);
```

TRİGGERLAR

1-)(Yeni bir yetişkin eklendiğinde, yasliMi sütununu otomatik olarak güncellemek için bir tetikleyici (trigger) Bu tetikleyici, yeni bir kayıt eklendiğinde veya mevcut bir kayıt güncellendiğinde yasliMi değerini otomatik olarak hesaplar.)

```
CREATE OR REPLACE FUNCTION yasliMiHesapla()
RETURNS TRIGGER AS $$
BEGIN
   NEW.yasliMi = yasHesapla(
   (SELECT DogumTarihi FROM Kisi WHERE Kisi.KisiID = NEW.KisiID)
   );
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER TriggerYasliMiHesapla
BEFORE INSERT OR UPDATE ON Yetiskin
FOR EACH ROW
EXECUTE FUNCTION yasliMiHesapla();
```

2-) Bu trigger, eğer randevu alan müşteri bir öğrenci ise, müşteri tipi 'Ogrenci' ise ödemenin tutarına %20 indirim uygular.

```
CREATE OR REPLACE FUNCTION OgrenciIndirimHesapla()
```

```
RETURNS TRIGGER AS $$
DECLARE
   musteriTipi VARCHAR;
   hizmetTutar MONEY;
   indirimliTutar MONEY;
BEGIN
   -- Müşteri tipini belirle
   SELECT Musteri.MusteriTipi
   INTO musteriTipi
   FROM Musteri
   WHERE Musteri.KisiID = (SELECT MusteriKisiID FROM Randevu WHERE
Randevu.RandevuID = NEW.RandevuID);
 -- Hizmet ücretini hesapla
   SELECT SUM(Hizmet.Ucret)
   INTO hizmetTutar
   FROM RandevuHizmet
   JOIN Hizmet ON RandevuHizmet.HizmetID = Hizmet.HizmetID
   WHERE RandevuHizmet.RandevuID = NEW.RandevuID;
-- Eğer müşteri tipi öğrenci ise indirimi uygula
   IF musteriTipi = 'Ogrenci' THEN
      indirimliTutar := hizmetTutar * 0.8; -- %20 indirim
   ELSE
      indirimliTutar := hizmetTutar; -- İndirim yok
   END IF;
 -- Ödeme tablosuna tutarı ekle
   NEW.Tutar := indirimliTutar;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER ogrenciIndirimTrigger
BEFORE INSERT ON Odeme
FOR EACH ROW
EXECUTE FUNCTION OgrenciIndirimHesapla();
```

3-) Odeme tablosuna yeni bir ödeme eklendiğinde, eğer randevu alan müşteri bir yetişkinse ve yasliMi alanı true ise ödeme tutarına %20 indirim uygular

```
CREATE OR REPLACE FUNCTION YasilMusteriIndirimHesapla()
RETURNS TRIGGER AS $$
DECLARE
   yasliMi BOOLEAN;
   hizmetTutar MONEY;
   indirimliTutar MONEY;
BEGIN
   -- Müşteri bilgilerini al
   SELECT y.yasliMi
   INTO yasliMi
   FROM Yetiskin y
   JOIN Randevu r ON r.MusteriKisiID = y.KisiID
   WHERE r.RandevuID = NEW.RandevuID;

   -- Hizmet ücretini hesapla
   SELECT SUM(Hizmet.Ucret)
   INTO hizmetTutar
   FROM RandevuHizmet
   JOIN Hizmet ON RandevuHizmet.HizmetID = Hizmet.HizmetID
   WHERE RandevuHizmet.RandevuID = NEW.RandevuID;
```

```
    -- Eğer yasliMi true ise %20 indirim uygula
    IF yasliMi = TRUE THEN
        indirimliTutar := hizmetTutar * 0.8; -- %20 indirim
    ELSE
        indirimliTutar := hizmetTutar; -- İndirim yok
    END IF;
    -- Ödeme tablosuna yeni tutarı yaz
    NEW.Tutar := indirimliTutar;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER yasliMusteriIndirimTrigger
BEFORE INSERT ON Odeme
FOR EACH ROW
EXECUTE FUNCTION YasilMusteriIndirimHesapla();
```

4-) Randevu ve hizmet arasındaki çok-çok ilişkiyi yönetmek için RandevuHizmet tablosuna otomatik kayıt eklemek amacıyla bir triggerdır. bir randevu kaydı yapıldığında ve hizmet seçildiğinde RandevuHizmet tablosuna otomatik olarak bir ilişki kaydı ekleyecektir.

```
CREATE OR REPLACE FUNCTION randevu_hizmet_ekle()
RETURNS TRIGGER AS
$$
BEGIN
    -- Eğer hizmetID null değilse, RandevuHizmet tablosuna ilişki ekleniyor
```

```
    IF NEW.hizmetID IS NOT NULL THEN
        INSERT INTO RandevuHizmet (hizmetID, randevuID)
        VALUES (NEW.hizmetID, NEW.randevuID);
    END IF;
    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER trg_randevu_hizmet_ekle
AFTER INSERT ON Randevu
FOR EACH ROW
EXECUTE FUNCTION randevu_hizmet_ekle();
```

5-) Aşağıdaki trigger, Musteri tablosuna musteriTipi sütunu 'Yetiskin' olan bir kayıt eklendiğinde, otomatik olarak Yetiskin tablosuna bu kişinin eklenmesini sağlar.

```
CREATE OR REPLACE FUNCTION musteriYetiskinEkle()
RETURNS TRIGGER AS $$
BEGIN
    -- Eğer müşteri tipi 'Yetiskin' ise
    IF NEW.musteriTipi = 'Yetiskin' THEN
        -- Yetiskin tablosuna kişi eklenir
        INSERT INTO Yetiskin (kisiID, yasliMi)
        VALUES (NEW.kisiID, NULL); -- yasliMi NULL olarak ayarlanır
    END IF;
    RETURN NEW; -- İşlemi tamamlayıp devam et
END;
```

```
$$ LANGUAGE plpgsql;


CREATE TRIGGER trg_musteri_yetiskin_ekle

AFTER INSERT ON Musteri

FOR EACH ROW

EXECUTE FUNCTION musteriYetiskinEkle();
```

FONKSİYONLAR

1-) EkipmanID'si girildiğinde, tüm şubelerdeki o ekipmandan toplam kaç adet bulunduğunu döndüren bir fonksiyondur. . Bu fonksiyon, SubeEkipman tablosunda EkipmanID'sine göre tüm adetleri toplayacaktır.

```
CREATE OR REPLACE FUNCTION toplam_ekipman_adet(ekipman_id INT)

RETURNS INT AS $$

DECLARE

  toplam_adet INT;

BEGIN

  -- EkipmanID'ye göre toplam adetleri hesapla

  SELECT SUM(adet) INTO toplam_adet

  FROM SubeEkipman

  WHERE EkipmanID = ekipman_id;


  -- Sonuç olarak toplam adeti döndür

  RETURN COALESCE(toplam_adet, 0); -- Eğer sonuç null ise 0 döndür

END;

$$ LANGUAGE plpgsql;
```

[SELECT toplam_ekipman_adet(1); -- EkipmanID'si 1 olan ekipmanın toplam adetini döndürür]

2-) Bu fonksiyon, verilen doğum tarihi parametresine göre, bir kişinin 60 yaşında veya daha büyük olup olmadığını kontrol eder ve buna göre TRUE ya da FALSE döndürür. (bu fonksiyonu TriggerYasliMiHesapla içindeki fonksiyon çağırıyor.)

```
CREATE OR REPLACE FUNCTION yasHesapla(dogumTarihi DATE)
RETURNS BOOLEAN AS $$
BEGIN
   RETURN EXTRACT(YEAR FROM AGE(CURRENT_DATE, dogumTarihi)) >= 60;
END;
$$ LANGUAGE plpgsql;
```

3-) Bu fonksiyon, verilen bir kisiID'ye göre kişinin doğum tarihini alır, bu tarihten yaşını hesaplar ve sonucu döndürür.

```
CREATE OR REPLACE FUNCTION yasHesaplaKisi(p_kisiID INT)
RETURNS INT AS $$
DECLARE
   kisiDogumTarihi DATE;  -- Değişken adını değiştirdik
   yas INT;
BEGIN
   -- Kisi tablosundan kişinin doğum tarihini alıyoruz
   SELECT dogumTarihi INTO kisiDogumTarihi
   FROM Kisi
   WHERE kisiID = p_kisiID;  -- Burada parametreyi p_kisiID olarak kullanıyoruz

   -- Eğer kişi bulunamazsa hata mesajı döndür
   IF kisiDogumTarihi IS NULL THEN
      RAISE EXCEPTION 'KisiID bulunamadı';
   END IF;
```

```
    -- Kişinin yaşını hesapla
    yas := EXTRACT(YEAR FROM AGE(CURRENT_DATE, kisiDogumTarihi));

    -- Yaşı döndür
    RETURN yas;
END;
$$ LANGUAGE plpgsql;
[SELECT yasHesaplaKisi(2);]
```

4-) Bir müşterinin tüm randevularını listelemek için bir fonksiyon. Randevu tablosunda musteriKisiID ile filtreleme yaparak, müşteriye ait tüm randevuları döndürür.

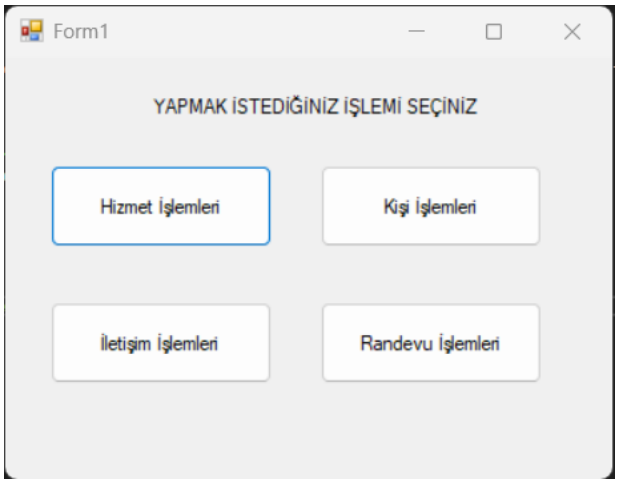
```
CREATE OR REPLACE FUNCTION musteriTumRandevular(p_musteriKisiID INT)
RETURNS TABLE (randevuID INT, berberKisiID INT, subeID INT, kampanyaID INT,
randevuTarihi TIMESTAMP) AS $$
BEGIN
    -- Müşterinin tüm randevularını döndürüyoruz
    RETURN QUERY
    SELECT r.randevuID, r.berberKisiID, r.subeID, r.kampanyaID, r.randevuTarihi
    FROM Randevu r
    WHERE r.musteriKisiID = p_musteriKisiID;
END;
$$ LANGUAGE plpgsql;

[SELECT * FROM musteriTumRandevular(3);]
```

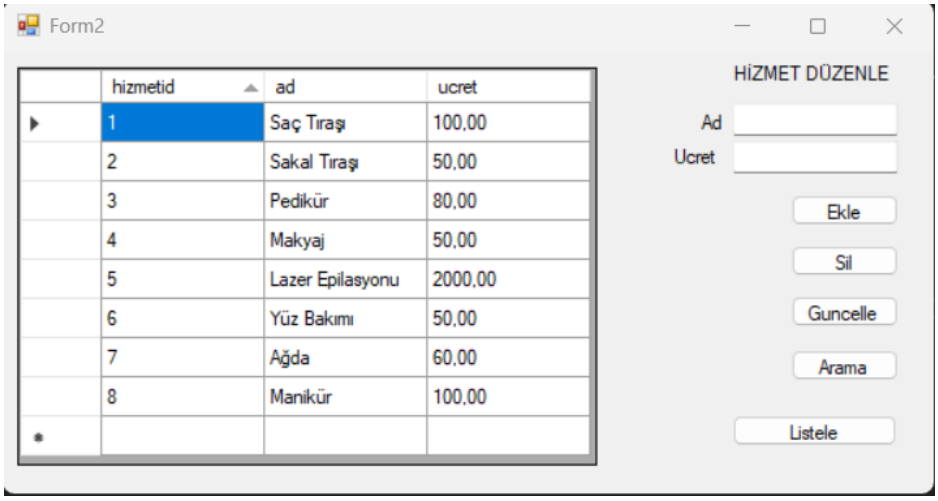
5-) Bu fonksiyon, bir kişinin kimlik numarasını (kisiID) alarak, o kişinin cinsiyet bilgisini döndürür.

```
CREATE OR REPLACE FUNCTION kisiCinsiyetiniOgren(p_kisiID INT)
RETURNS VARCHAR AS $$
DECLARE
   kisiCinsiyet VARCHAR;
BEGIN
   -- Kisi tablosundan kişinin cinsiyetini alıyoruz
   SELECT cinsiyet INTO kisiCinsiyet
   FROM Kisi
   WHERE kisiID = p_kisiID;

   -- Eğer kişi bulunamazsa hata mesajı döndür
   IF kisiCinsiyet IS NULL THEN
      RAISE EXCEPTION 'KisiID bulunamadı veya cinsiyet bilgisi yok';
   END IF;

   -- Cinsiyeti döndür
   RETURN kisiCinsiyet;
END;
$$ LANGUAGE plpgsql;
```

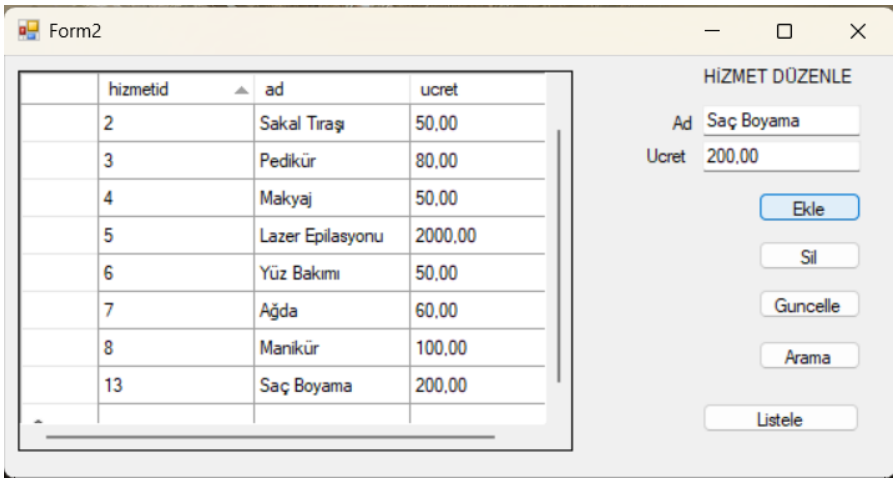
[SELECT kisiCinsiyetiniOgren(2);]

Uygulama Ekran Görüntüleri

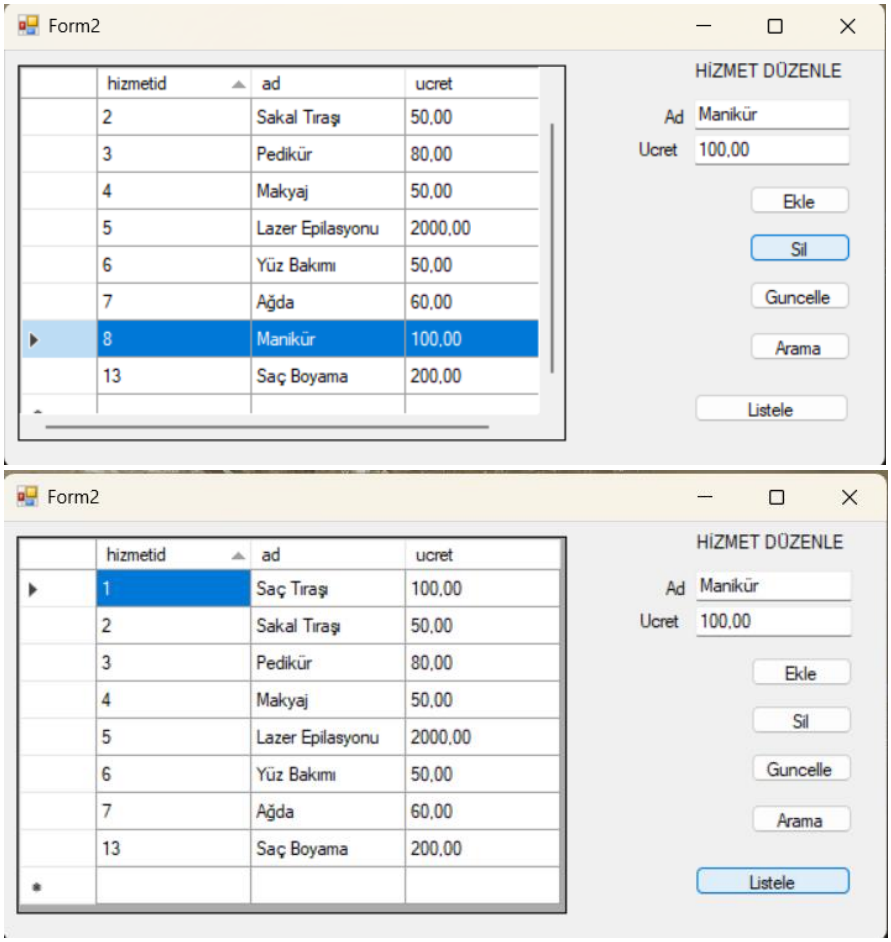




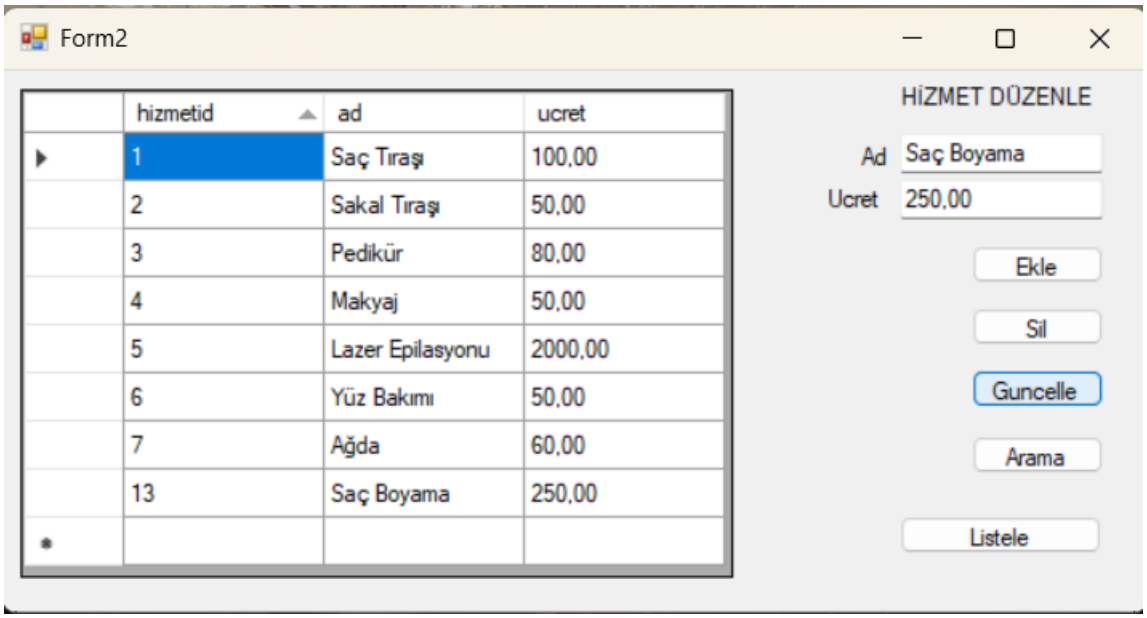
Ekleme İşlemi(13 nolu hizmet eklendi)



Silme İşlemi(8 nolu hizmet silindi)



Güncelleme İşlemi(13 nolu hizmet ucreti güncellendi)



Arama İşlemi(13 nolu hizmet arandı)