SAKARYA
ÜNİVERSİTESİ

Öğrenci Adı: Münevver Sude

Öğrenci Soyadı: İSTEKLİ

Öğrenci Numarası: G231210031

Öğrenci E-postası: munevver.istekli@ogr.sakarya.edu.tr

Öğrenci Adı: Ayşe

Öğrenci Soyadı: TAŞKAN

Öğrenci Numarası: G231210043

Öğrenci E-postası: ayse.taskan@ogr.sakarya.edu.tr

Uygulama Kaynak Kodları GitHub Linki :

aystskan/VTYS_PROJE: 202425_GÜZ_VTYS _PROJE_TARIM

Uygulamanın Tanımı: Tarımla ilgilenen bireysel ve kurumsal kişilerin ihtiyaçları olan kontrolleri ve zirai ürün takip etmeyi sağlayan bir program.

İş Kuralları:

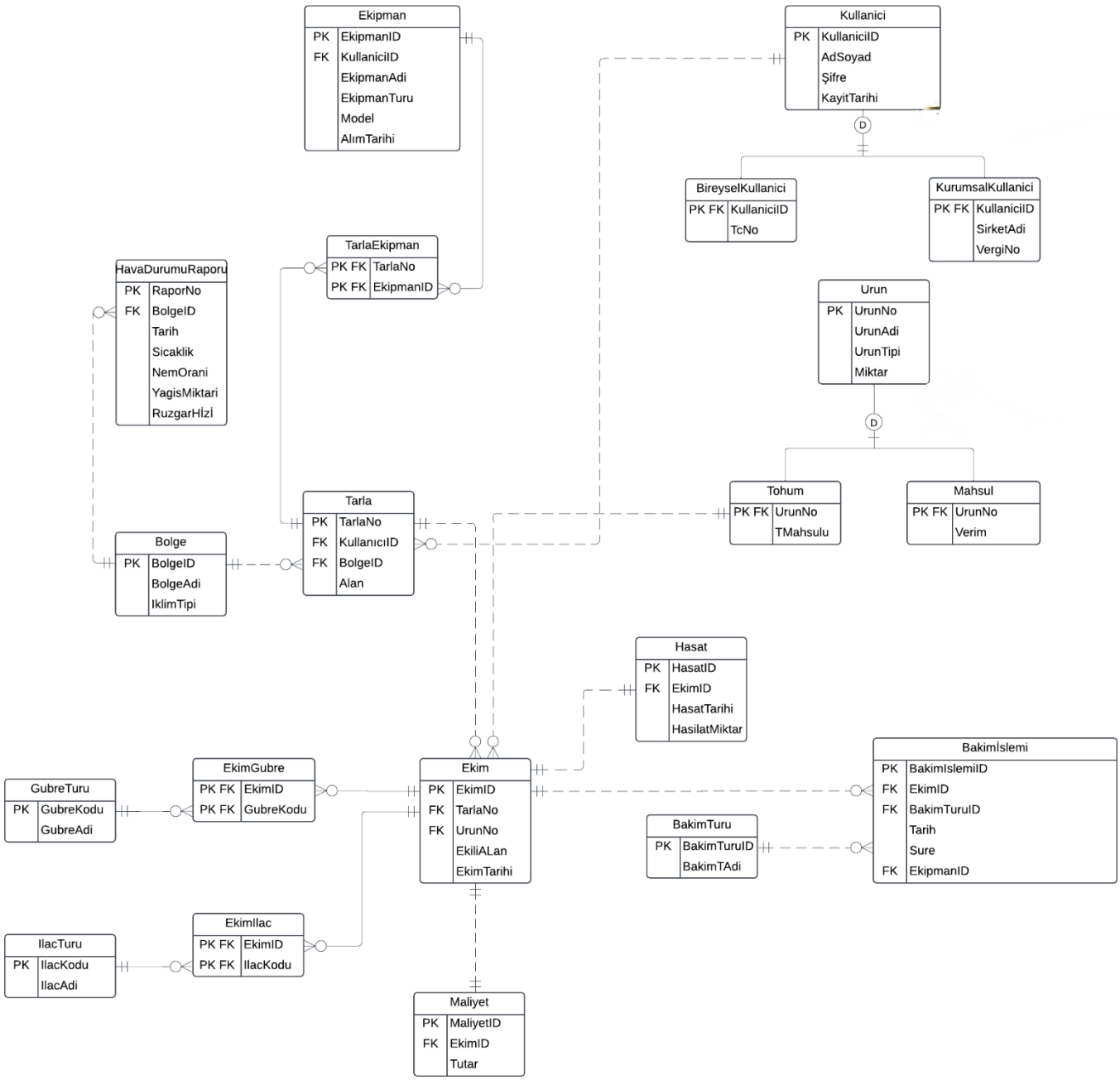
- Bu veritabanında her kullanıcı için benzersiz bir KullanıcıID olmalıdır. AdSoyad, şifre ve kayıt tarihi tutulmalıdır
- Bireysel Kullanıcı tablosunda, her kullanıcı için TCNo benzersiz olmalıdır. Kurumsal Kullanıcı tablosunda, her kullanıcı için VergiNo benzersiz olmalıdır, SirketAdi gereklidir ve boş olamaz.
- Tarla tablosunda her tarla TarlaID olmalıdır.
- Bölge tablosunda her bölge BölgeID olmalıdır. Her Bölge, bir BölgeAdi ve IklimTipi ile tanımlanmalıdır.
- Urun tablosunda her ürün için bir UrunNo olmalıdır. Her ürünün bir UrunAdi, UrunTipi (örneğin, sebze, meyve vb.) ve Miktar (stok miktarı) bilgisi olmalıdır.
- Ekipman tablosunda her ekipman için benzersiz bir EkipmanID olmalıdır.Her ekipmanın EkipmanAdi, EkipmanTuru (örneğin, sulama cihazı, traktör vb.), Model ve AlimTarihi bilgileri olmalıdır.
- HavaDurumuRaporu tablosunda her hava durumu raporu için benzersiz bir RaporNo olmalıdır. Sicaklik, NemOrani, YagisMiktari ve RuzgarHizi bilgilerini içermelidir.
- Ekim tablosunda her ekim işlemi için benzersiz bir EkimID olmalıdır. Ekim tablosu, EkilenAlan ve EkimTarihi gibi bilgileri içermelidir.
- Hasat tablosunda her hasat için benzersiz bir HasatID olmalıdır. HasatTarihi ve HasatMiktarı bilgileri bulunmalıdır.
- Maliyet tablosunda her maliyet işlemi için benzersiz bir MaliyetID olmalıdır. Tabloda tutar bilgisi bulunmalıdır.
- BakımTuru tablosunda her bakım turu için benzersiz bir BakımTuruID olmalıdır. BakımİslemiID tablosunda her bakım işlemi için benzersiz bir BakımİslemiID olmalıdır. BakımİslemiID Tarih, Sure ve EkipmanID bilgilerini içermelidir.
- GübreTuru tablosunda her gübre türü için benzersiz bir GübreTuruID olmalıdır her gübre türünün GübreAdı bilgisi bulunmalıdır.
- Her bireysel kullanıcı yalnızca 1 TC'ye sahip olabilir, 1 TC'nin yalnızca 1 sahibi olabilir.
- Her kurumsal kullanıcı yalnızca 1 şirket adı ve vergi no'ya sahip olabilir. Bir şirket adı ve vergi numarası en az 1 k.k'ye ait, en fazla birden çok k.k'ye ait olabilir.
- Bir kullanıcı hiçbir tarla kaydetmeyeceği gibi bir veya daha fazla da kaydedebilir, bir tarlanın yalnızca 1 kullanıcısı olabilir.
- Bir tarlada hiç ürün olmayacağı gibi birden çok ürün de yetiştirilebilir
- Bir ekimi yalnızca 1 tarlaya ait olabilir, bir tarlada hiç ya da birçok ekim olabilir.
- Bir tohum hiçbir ekimde yer almayacağı gibi birden çok ekimle ilişkilendirilebilir. Bir ekimde yalnızca bir ürün ekilebilir.
- Bir tarla yalnızca 1 bölgede olabilir. Bir bölge birden fazla tarlayı içerebileceği gibi hiçbir tarlayı içermeyebilir
- Bir ürün için hiçbir bakım işlemi yapılmamış olabileceği gibi birden çok bakım işlemi yapılmış olabilir. Bir bakım işlemi yalnızca bir ekimde yapılabilir.

- Bir bakım türü hiçbir bakım işleminde kullanılmayabileceği gibi aynı zamanda birden çok bakım işleminde kullanılabilir. Bir bakım işlemi yalnızca bir bakım türüne aittir.
- Bir ekimin yalnızca 1 hasatı olabilir, bir hasat yalnızca bir ekimindir.
- Her hava durumu yalnızca bir bölgede geçerlidir, bir bölgede hiç hava durumu kaydedilmemiş olabilir her bölge için birden fazla hava durumu da kaydedilebilir.
- Bir tarlada birden fazla ekipman ya da hiç ekipman kullanımı olabilir. Bir ekipman hiç ya da birden fazla tarlada kullanılabilir.
- Bir ekimde en fazla çok gübre Türü kullanıldığı gibi en az 1 gübreT kullanılır. Bir gübre hiç ya da birden çok ekimde kullanılabilir.
- Bir ekimde hiç ilaçT kullanılmayabileceği gibi birden çok ilaçT kullanılabilir. Bir ilaçT hiç ekimde kullanılmadığı gibi birden çok ekimde de kullanılmış olabilir.
- Bir maliyet kaydı yalnızca 1 ekime aittir. Bir ekimin yalnızca 1 maliyet kaydı vardır.

İlişkisel Şema:

- Kullanıcı(KullanıcıID serial, AdSoyad varchar)
- Bireysel Kullanıcı(KullanıcıID int,TCNo char)
- Kurumsal Kullanıcı(KullanıcıID int , VergiNo char, SirketAdi varchar)
- Tarla(TarlaID serial, KullanıcıID int, BölgeID int, Alan numeric)
- Bolge(BölgeID serial, BölgeAdi varchar, IklimTipi varchar)
- Urun (UrunNo serial, UrunAdi varchar, UrunTipi varchar, Miktar int)
- Tohum(UrunNo int, TMahsulu varchar)
- Mahsul(UrunNo int, Verim numeric)
- Ekipman(EkipmanID serial, KullanıcıID int, EkipmanAdi varchar, EkipmanTuru varchar, Model varchar, AlimTarihi date)
- TarlaEkipman(TarlaNo int, EkipmanID int)
- HavaDurumuRaporu(RaporNo serial, BölgeID int, Tarih date, Sicaklik numeric, NemOrani numeric, YagisMiktari numeric, RuzgarHizi numeric)
- Ekim(EkimID serial, TarlaNo int, UrunNo int, EkilenAlan numeric, EkimTarihi date)
- EkimGubre(EkimID int, GubreKodu int)
- EkimIlac(EkimID int, IlacKodu int)
- Hasat(HasatID serial, EkimID int, HasatTarihi date, HasatMiktarı numeric)
- Maliyet(MaliyetID serial, EkimID int, Tutar numeric)
- BakımTuru(BakımTuruID serial, BakımAdı varchar)
- BakımİslemI(BakımİslemiID serial, EkimID int, BakımTuruID int, Tarih date, Sure numeric, EkipmanID int)
- GübreTuru(GübreTuruID serial, GübreAdı varchar)
- IlacTuru(IlacKodu serial, IlacAdı varchar)

Varlık Bağıntı Modeli:



Fonksiyonlar (Stored Procedures/Functions):

- Kullanıcı Ekleme Fonksiyonu:

Yeni bir kullanıcı ekler. Kullanıcı türüne (bireysel veya kurumsal) göre ilgili tabloya T.C. kimlik numarası veya şirket bilgilerini kaydeder.

- Ürün Güncelleme Fonksiyonu:

Belirtilen ürün numarasına ait ürünün adını, tipini ve miktarını günceller.

- Ekipman Silme Fonksiyonu:

Verilen ekipman ID’sine göre ilgili ekipmanı veritabanından siler.

- Ekim Alanı Arama Fonksiyonu:

Belirli bir minimum ve maksimum alana sahip ekim kayıtlarını arar ve sonuçları döner.

Tetikleyiciler (Triggers):

- Kayıt Tarihi Ekleme:

Yeni bir kullanıcı eklenirken, kayıt tarihini otomatik olarak bugünün tarihiyle doldurur.

- Ürün Silme:

Bir ürün silindiğinde, bu ürüne bağlı ekim kayıtlarını otomatik olarak temizler.

- Güncelleme Tarihi Ekleme:

Ekipman bilgileri güncellendiğinde, güncelleme tarihini otomatik olarak bugünün tarihiyle kaydeder.

- Ekim Alanı Güncelleme:

Ekim tablosundaki ekim alanı değiştiğinde, otomatik olarak belirli bir alana sahip kayıtları arar.

SQL İFADELERİ:

```
CREATE TABLE kullanici (
    "KullaniciID" SERIAL,
    "AdSoyad" VARCHAR(100),
    "Sifre" VARCHAR(100),
    "KayitTarihi" DATE,
    CONSTRAINT "kullaniciPK" PRIMARY KEY ("KullaniciID")
);
CREATE TABLE "BireyselKullanici" (
    "KullaniciID" INT,
    "TcNo" VARCHAR(11) UNIQUE NOT NULL,
    CONSTRAINT "BireyselKullanici_PK" PRIMARY KEY ("KullaniciID"),
    CONSTRAINT "FK_BireyselKullanici" FOREIGN KEY ("KullaniciID")
        REFERENCES "kullanici" ("KullaniciID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE "KurumsalKullanici" (
    "KullaniciID" INT,
    "VergiNo" VARCHAR(15) UNIQUE,
    "SirketAdi" VARCHAR(150),
```

```
    CONSTRAINT "KurumsalKullanici_PK" PRIMARY KEY ("KullaniciID"),
    CONSTRAINT "FK_KurumsalKullanici" FOREIGN KEY ("KullaniciID")
    REFERENCES "kullanici" ("KullaniciID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE urun(
    "UrunNo" SERIAL,
    "UrunAdi" VARCHAR(255) NOT NULL,
    "UrunTipi" VARCHAR(50),
    "Miktar" INT,
    CONSTRAINT "UrunPK" PRIMARY KEY ("UrunNo")
);
CREATE TABLE "Tohum"(
    "UrunNo" INT,
    "TMahsulu" VARCHAR(255),
    CONSTRAINT "TohumPK" PRIMARY KEY ("UrunNo"),
    CONSTRAINT "FKTohum" FOREIGN KEY ("UrunNo")
    REFERENCES "urun" ("UrunNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE "Mahsul"(
    "UrunNo" INT,
    "Verim" Numeric,
    CONSTRAINT "MahsulPK" PRIMARY KEY ("UrunNo"),
    CONSTRAINT "FKMahsul" FOREIGN KEY ("UrunNo")
    REFERENCES "urun" ("UrunNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE ekipman(
```

```
    "EkipmanID" SERIAL,
    "KullaniciID" INT,
    "EkipmanAdi" VARCHAR(50) NOT NULL,
    "EkipmanTuru" VARCHAR(50) NOT NULL,
    "Model" VARCHAR(50),
    "AlimTarihi" DATE,
    CONSTRAINT "EkipmanPK" PRIMARY KEY ("EkipmanID"),
    CONSTRAINT "FKEkipman" FOREIGN KEY ("KullaniciID")
    REFERENCES "kullanici" ("KullaniciID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE bolge(
    "BolgeID" SERIAL,
    "BolgeAdi" VARCHAR(50),
    "IklimTipi" VARCHAR(50),
    CONSTRAINT "BolgePK" PRIMARY KEY ("BolgeID")
);
CREATE TABLE Tarla(
    "TarlaNo" SERIAL,
    "KullaniciID" INT,
    "BolgeID" INT,
    "Alan" NUMERIC(10,2) NOT NULL,
    CONSTRAINT "TarlaPK" PRIMARY KEY ("TarlaNo"),
    CONSTRAINT "FKTarla" FOREIGN KEY ("BolgeID")
    REFERENCES "bolge" ("BolgeID")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "FKTarlaKul" FOREIGN KEY ("KullaniciID")
    REFERENCES "kullanici" ("KullaniciID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
```

```
    );
CREATE TABLE TarlaEkipman(
    "TarlaNo" INT,
    "EkipmanID" INT,
    CONSTRAINT "TarlaEkipmanPK" PRIMARY KEY ("TarlaNo", "EkipmanID"),
    CONSTRAINT "FKTarlaEkipman" FOREIGN KEY ("TarlaNo")
        REFERENCES "tarla" ("TarlaNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "FKTarlaEkipman_Ekipman" FOREIGN KEY ("EkipmanID")
        REFERENCES "ekipman" ("EkipmanID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE HavaDurumuRaporu(
    "RaporNo" SERIAL,
    "BolgeID" INT,
    "Tarih" DATE,
    "Sicaklik" NUMERIC(5,2),
    "NemOrani" NUMERIC(5,2),
    "YagisMiktari" NUMERIC(5,2),
    "RuzgarHizi" NUMERIC(5,2),
    CONSTRAINT "HavaDurumuPK" PRIMARY KEY ("RaporNo"),
    CONSTRAINT "FKHavaDurumu" FOREIGN KEY ("BolgeID")
        REFERENCES "bolge" ("BolgeID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE Ekim(
    "EkimID" SERIAL,
    "TarlaNo" INT,
    "UrunNo" INT NOT NULL,
```

```
    "EkilenALan" NUMERIC(10,2),
    "EkimTarihi" DATE NOT NULL,
    CONSTRAINT "EkimPK" PRIMARY KEY ("EkimID"),
    CONSTRAINT "FKEkim" FOREIGN KEY ("TarlaNo")
        REFERENCES "tarla" ("TarlaNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "EkimFK" FOREIGN KEY ("UrunNo")
        REFERENCES "urun" ("UrunNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE Hasat(
    "HasatID" SERIAL,
    "EkimID" INT,
    "HasatTarihi" DATE,
    "HasilatMiktarı" NUMERIC,
    CONSTRAINT "HasatPK" PRIMARY KEY ("HasatID"),
    CONSTRAINT "FKHasat" FOREIGN KEY ("EkimID")
        REFERENCES "ekim" ("EkimID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE Maliyet(
    "MaliyetID" SERIAL,
    "EkimID" INT,
    "Tutar" Numeric(10,2) NOT NULL,
    CONSTRAINT "MaliyetPK" PRIMARY KEY ("MaliyetID"),
    CONSTRAINT "FKMAliyet" FOREIGN KEY ("EkimID")
        REFERENCES "ekim" ("EkimID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
```

```
);
CREATE TABLE BakimTuru(
    "BakimTuruID" SERIAL,
    "BakimTAdi" VARCHAR(100) NOT NULL,
    CONSTRAINT "BakimTuruPK" PRIMARY KEY ("BakimTuruID")
);
CREATE TABLE BakimIslemi(
    "BakimIslemiID" SERIAL,
    "EkimID" INT,
    "BakimTuruID" INT,
    "Tarih" DATE,
    "Sure" NUMERIC(5,2),
    "EkipmanID" INT,
    CONSTRAINT "BakimIslemiPK" PRIMARY KEY ("BakimIslemiID"),
    CONSTRAINT "FKBakimIslemi" FOREIGN KEY ("EkimID")
        REFERENCES "ekim" ("EkimID")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "BakimIslemi_FK" FOREIGN KEY ("BakimTuruID")
        REFERENCES "bakimturu" ("BakimTuruID")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "BakimIslemiFK" FOREIGN KEY ("EkipmanID")
        REFERENCES "ekipman" ("EkipmanID")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE GubreTuru(
    "GubreKodu" SERIAL,
    "GubreAdi" VARCHAR(50),
    CONSTRAINT "GubreTuruPK" PRIMARY KEY ("GubreKodu")
);
```

```
CREATE TABLE IlacTuru(
    "IlacKodu" SERIAL,
    "IlacAdi" VARCHAR(50),
    CONSTRAINT "IlacTuruPK" PRIMARY KEY ("IlacKodu")
);
CREATE TABLE EkimGubre(
    "EkimID" INT,
    "GubreKodu" INT,
    CONSTRAINT "EkimGubrePK" PRIMARY KEY ("EkimID", "GubreKodu"),
    CONSTRAINT "FKEkimGubre_Ekim" FOREIGN KEY ("EkimID")
        REFERENCES "ekim" ("EkimID")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "FKEkimGubre_Gubre" FOREIGN KEY ("GubreKodu")
        REFERENCES "gubreturu" ("GubreKodu")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
CREATE TABLE EkimIlac(
    "EkimID" INT,
    "IlacKodu" INT,
    CONSTRAINT "EkimIlacPK" PRIMARY KEY ("EkimID", "IlacKodu"),
    CONSTRAINT "FKEkimIlac_Ekim" FOREIGN KEY ("EkimID")
        REFERENCES "ekim" ("EkimID")
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CONSTRAINT "FKEkimIlac_Ilac" FOREIGN KEY ("IlacKodu")
        REFERENCES "ilacturu" ("IlacKodu")
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
--kullanıcı ekleme fonksiyonu
```

```
CREATE OR REPLACE FUNCTION kullanici_ekle(
p_adsoyad VARCHAR,    p_sifre VARCHAR,    p_tur
VARCHAR, --  'bireysel'  veya 'kurumsal'        p_tcno
VARCHAR   DEFAULT   NULL,              p_vergino
VARCHAR   DEFAULT   NULL,             p_sirketadi
VARCHAR DEFAULT NULL
) RETURNS VOID AS $$
DECLARE
   v_kullaniciid INT; -- Kullanıcı ID'sini tutmak için değişken tanımlanıyor
BEGIN
   -- Genel kullanıcı ekle
   INSERT INTO kullanici ("AdSoyad", "Sifre", "KayitTarihi")
   VALUES (p_adsoyad, p_sifre, CURRENT_DATE)
   RETURNING "KullaniciID" INTO v_kullaniciid;
   -- Türüne göre bireysel veya kurumsal kullanıcı ekle
   IF p_tur = 'bireysel' THEN
      INSERT INTO "BireyselKullanici" ("KullaniciID", "TcNo")
      VALUES (v_kullaniciid, p_tcno);
   ELSIF p_tur = 'kurumsal' THEN
      INSERT INTO "KurumsalKullanici" ("KullaniciID", "VergiNo", "SirketAdi")
      VALUES (v_kullaniciid, p_vergino, p_sirketadi);
   END IF;
END;
$$ LANGUAGE plpgsql;
--ürün güncelleme fonksiyonu
CREATE OR REPLACE FUNCTION urun_guncelle(
   p_urunno INT,
p_urunadi VARCHAR,
p_uruntipi VARCHAR,
p_miktar INT
) RETURNS VOID AS $$
BEGIN
```

```
    UPDATE urun
    SET "UrunAdi" = p_urunadi,
        "UrunTipi" = p_uruntipi,
        "Miktar" = p_miktar
    WHERE "UrunNo" = p_urunno;
END;
$$ LANGUAGE plpgsql;
--ekipman silmek için fonksiyon
CREATE OR REPLACE FUNCTION ekipman_sil(p_ekipmanid INT) RETURNS VOID AS $$
BEGIN
    DELETE FROM ekipman -- Burada tablo ismi küçük harfle yazılmalı
    WHERE "EkipmanID" = p_ekipmanid;
END;
$$ LANGUAGE plpgsql;
--ekim alanı aramak için fonksiyon
CREATE OR REPLACE FUNCTION ekim_alani_ara(
p_min_alan NUMERIC(10,2),     p_max_alan
NUMERIC(10,2)
) RETURNS TABLE(
    "EkimID" INT,
    "TarlaNo" INT,
    "UrunNo" INT,
    "EkilenALan" NUMERIC(10,2),
    "EkimTarihi" DATE
) AS $$
BEGIN
    RETURN QUERY
    SELECT "EkimID", "TarlaNo", "UrunNo", "EkilenALan", "EkimTarihi"
    FROM "Ekim"
    WHERE "EkilenALan" BETWEEN p_min_alan AND p_max_alan;
END;
$$ LANGUAGE plpgsql;
```

```
--yeni kayıta bugunun tarihini atmak icin trigger
CREATE OR REPLACE FUNCTION set_kayittarihi()
RETURNS TRIGGER AS $$
BEGIN
    NEW."KayitTarihi" := CURRENT_DATE;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_set_kayittarihi
BEFORE INSERT ON kullanici
FOR EACH ROW
EXECUTE FUNCTION set_kayittarihi();
--urun silme trigger
CREATE OR REPLACE FUNCTION urun_sil_ekim()
RETURNS TRIGGER AS $$
BEGIN
    DELETE FROM ekim WHERE "UrunNo" = OLD."UrunNo";
    RETURN OLD;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trg_urun_sil_ekim
AFTER DELETE ON urun
FOR EACH ROW
EXECUTE FUNCTION urun_sil_ekim();
--güncelleme trigger
CREATE OR REPLACE FUNCTION set_guncelleme_tarihi()
RETURNS TRIGGER AS $$
BEGIN
    NEW."GuncellemeTarihi" := CURRENT_DATE;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
ALTER TABLE ekipman ADD COLUMN "GuncellemeTarihi" DATE;

CREATE TRIGGER trg_set_guncelleme_tarihi

BEFORE UPDATE ON ekipman

FOR EACH ROW

EXECUTE FUNCTION set_guncelleme_tarihi();

--alan trigger

CREATE OR REPLACE FUNCTION ekim_alani_ara_trigger()

RETURNS TRIGGER AS $$

BEGIN

    -- Ekim alanı güncellenince, fonksiyonu tetikleyerek arama yapılabilir

    PERFORM ekim_alani_ara(0, 10000); -- örneğin, 0 ile 10000 arasındaki alanları arıyoruz

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Ekim tablosunda ekim alanı güncellendiğinde trigger tetiklenir

CREATE TRIGGER trg_ekim_alani_ara

AFTER UPDATE ON Ekim

FOR EACH ROW

WHEN (NEW."EkilenALan" <> OLD."EkilenALan") -- sadece alan değiştiğinde tetiklenecek EXECUTE
FUNCTION ekim_alani_ara_trigger();

-- Bireysel kullanıcı eklemek

SELECT kullanici_ekle(

    'Ahmet Yılmaz',

    'ahmet123',

    'bireysel',

    '12345678901',

    NULL,

    NULL

);

-- Kurumsal kullanıcı eklemek

SELECT kullanici_ekle(

    'XYZ A.Ş.',
```

```
    'xyz123',
    'kurumsal',
    NULL,
    '9876543210',
    'XYZ Agriculture'
);
INSERT INTO urun ("UrunNo","UrunAdi", "UrunTipi", "Miktar")
VALUES
    (1, 'Domates', 'Sebze', 100),
    (2, 'Biber', 'Sebze', 150),
    (3, 'Patates', 'Kök', 200);
INSERT INTO ekipman ("KullaniciID", "EkipmanAdi", "EkipmanTuru", "Model", "AlimTarihi")
VALUES
    (6, 'Traktör', 'Tarım', 'John Deere', '2022-05-10'),
    (7, 'Büyük Çapa Makinesi', 'Tarım', 'Kubota', '2023-07-20');
INSERT INTO tarla ("KullaniciID", "BolgeID", "Alan")
VALUES
    (6, 1, 200.50),
(7, 2, 300.00);
INSERT INTO ekim ("TarlaNo", "UrunNo", "EkilenALan", "EkimTarihi")
VALUES
    (1, 1, 150.00, '2024-03-10'),
(2, 2, 100.00, '2024-04-15');
SELECT urun_guncelle(1, 'Patates', 'Kök', 250);
SELECT * from ekipman
-- Ekipman güncelleme fonksiyonu ile
SELECT ekipman_guncelle(1, 'Yeni Traktör', 'Tarım', 'New John Deere', '2024-01-01');
SELECT * FROM ekipman WHERE "EkipmanID" = 1;
--TABLOLARI BİRLEŞTİRME İŞLEMLERİ
SELECT
    U."UrunNo",
    U."UrunAdi",
```

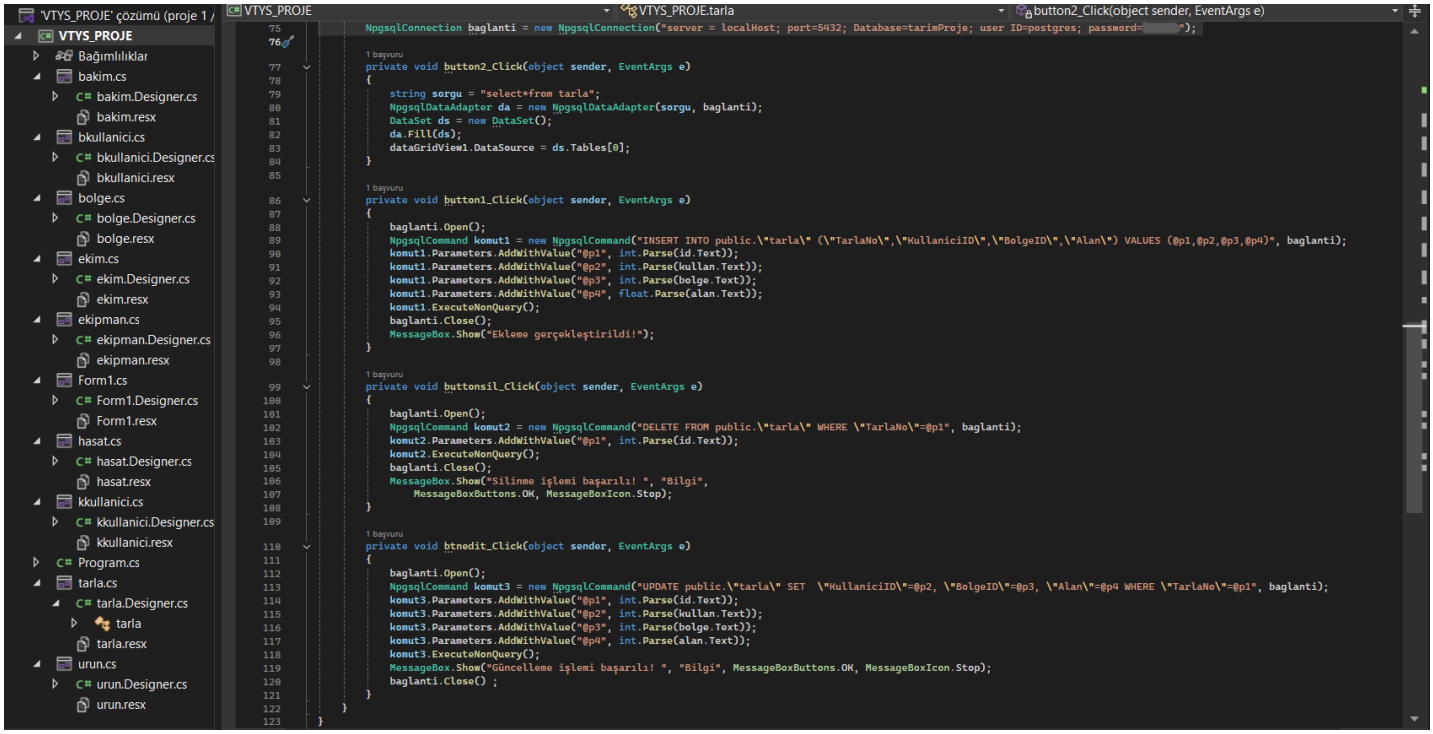
```
    U."UrunTipi",
    U."Miktar",
    T."TMahsulu" AS "TohumMahsulTuru",
    M."Verim"
FROM
    "urun" U
LEFT JOIN
    "Tohum" T ON U."UrunNo" = T."UrunNo"
LEFT JOIN
    "Mahsul" M ON U."UrunNo" = M."UrunNo";
----------------------------------------
SELECT
    K."KullaniciID" AS "ID",
    K."AdSoyad" AS "Ad Soyad",
    K."Sifre",
    K."KayitTarihi" AS "Kayıt Tarihi",
KK."SirketAdi" AS "Şirket Adı",
    KK."VergiNo" AS "Vergi No"
FROM
    "kullanici" K
LEFT JOIN
    "KurumsalKullanici" KK ON K."KullaniciID" = KK."KullaniciID";
-------------------------------------------
SELECT
    e."EkimID",
    h."HasatID",
    h."HasatTarihi",
    h."HasilatMiktarı"
FROM
    "ekim" e
LEFT JOIN
    "hasat" h
```

```
ON
    e."EkimID" = h."EkimID";
-----------------------------------------
SELECT
    e."EkipmanID",
    e."KullaniciID",
    k."AdSoyad" AS "Kullanıcı Adı",
    e."EkipmanAdi" AS "Ekipman Adı",
    e."EkipmanTuru" AS "Tür",
    e."Model",
    e."AlimTarihi" AS "Alım Tarihi"
FROM
    "ekipman" e
LEFT JOIN
    "kullanici" k
ON
    e."KullaniciID" = k."KullaniciID";
----------------------------------------
SELECT
    e."EkimID" AS "ID",
    e."TarlaNo" AS "Tarla No",
    e."UrunNo" AS "Ürün No",
    e."EkilenALan" AS "Ekili Alan",
    e."EkimTarihi" AS "Ekim Tarihi",
eg."GubreKodu" AS "Gübre", ei."IlacKodu" AS
"İlaç",

    m."Tutar" AS "Maliyet"
FROM
    "ekim" e
LEFT JOIN
    "ekimgubre" eg
```

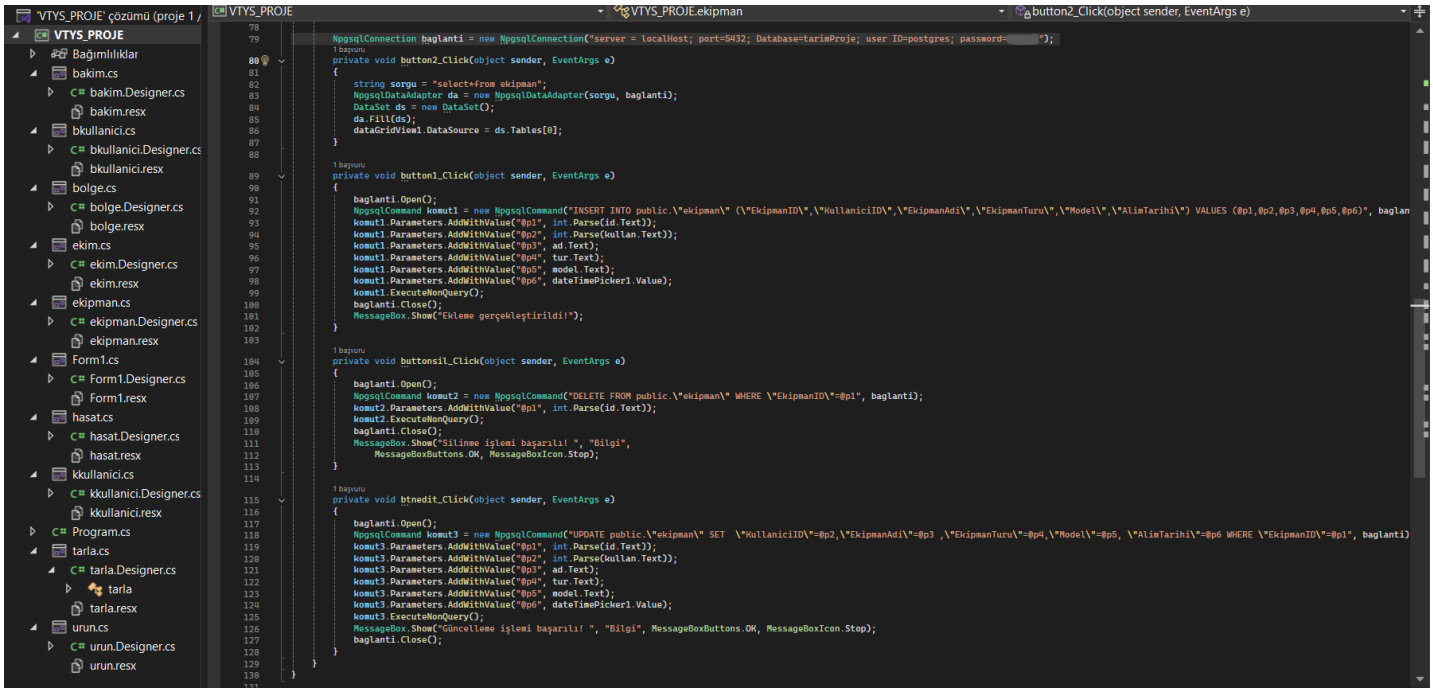
```
ON
    e."EkimID" = eg."EkimID"
LEFT JOIN
    "ekimilac" ei
ON
    e."EkimID" = ei."EkimID"
LEFT JOIN
    "maliyet" m
ON
    e."EkimID" = m."EkimID";
-----------------------------------
CREATE TEMP TABLE "BireyselTablo" AS
SELECT
    k."KullaniciID",
    k."AdSoyad",
    k."Sifre",
    k."KayitTarihi",
    b."TcNo"
FROM
    "kullanici" k
LEFT JOIN
    "BireyselKullanici" b
ON
    k."KullaniciID" = b."KullaniciID";
```

ARAYÜZ ÖRNEK GÖSTERİMLERİ

SQL bağlantıları arayüz ortamına kurulduktan sonra “tarla” ve “ekipman” tablolarımızdan ödev dosyasında istenen işlemler [arama(+listelendi), ekleme, silme, güncelleme] in gerçekleştirilmesi için yazılmış olan kod parçası örnekleri:

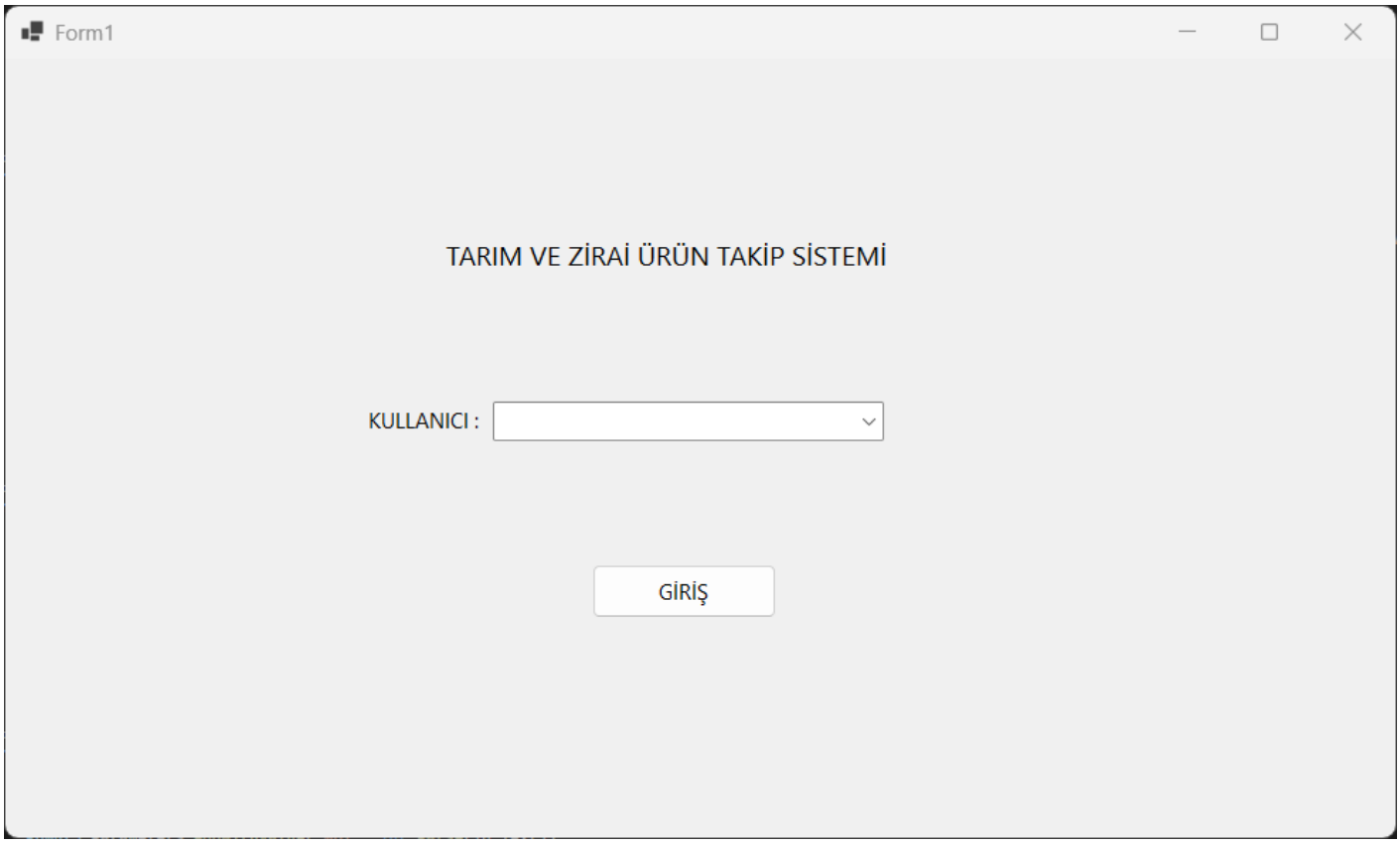


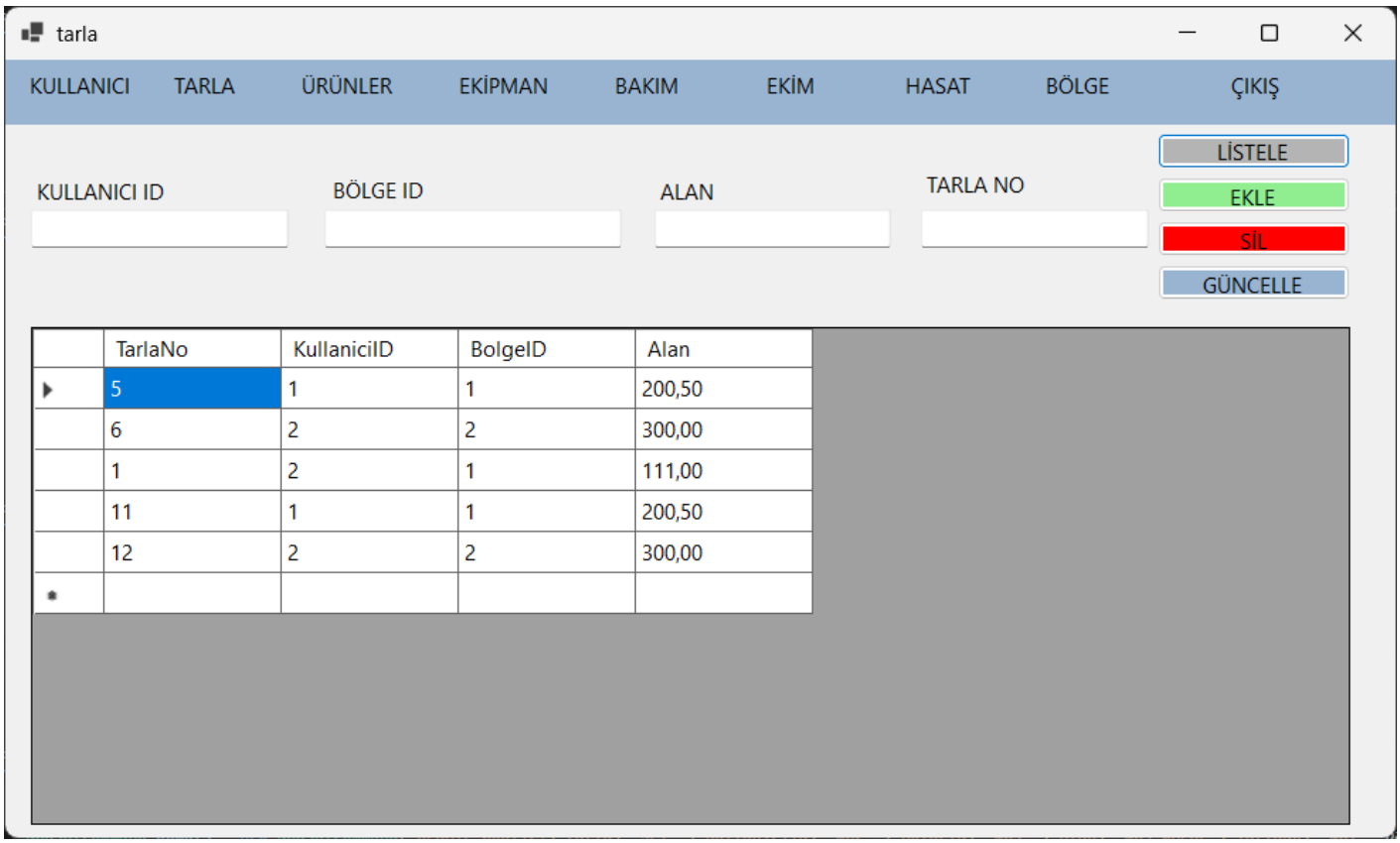
```
NpgsqlConnection baglanti = new NpgsqlConnection("server = localHost; port=5432; Database=tarimProje; user ID=postgres; password=[redacted]");

1 başvuru
private void button2_Click(object sender, EventArgs e)
{
    string sorgu = "select*from tarla";
    NpgsqlDataAdapter da = new NpgsqlDataAdapter(sorgu, baglanti);
    DataSet ds = new DataSet();
    da.Fill(ds);
    dataGridView1.DataSource = ds.Tables[0];
}

1 başvuru
private void button1_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komut1 = new NpgsqlCommand("INSERT INTO public.\"tarla\" (\"TarlaNo\",\"KullaniciID\",\"BolgeID\",\"Alan\") VALUES (@p1,@p2,@p3,@p4)", baglanti);
    komut1.Parameters.AddWithValue("@p1", int.Parse(id.Text));
    komut1.Parameters.AddWithValue("@p2", int.Parse(kullan.Text));
    komut1.Parameters.AddWithValue("@p3", int.Parse(bolge.Text));
    komut1.Parameters.AddWithValue("@p4", float.Parse(alan.Text));
    komut1.ExecuteNonQuery();
    baglanti.Close();
    MessageBox.Show("Ekleme gerçekleştirildi!");
}

1 başvuru
private void buttonsil_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komut2 = new NpgsqlCommand("DELETE FROM public.\"tarla\" WHERE \"TarlaNo\"=@p1", baglanti);
    komut2.Parameters.AddWithValue("@p1", int.Parse(id.Text));
    komut2.ExecuteNonQuery();
    baglanti.Close();
    MessageBox.Show("Silinme işlemi başarılı! ", "Bilgi",
        MessageBoxButtons.OK, MessageBoxIcon.Stop);
}

1 başvuru
private void btnedit_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komut3 = new NpgsqlCommand("UPDATE public.\"tarla\" SET  \"KullaniciID\"=@p2, \"BolgeID\"=@p3, \"Alan\"=@p4 WHERE \"TarlaNo\"=@p1", baglanti);
    komut3.Parameters.AddWithValue("@p1", int.Parse(id.Text));
    komut3.Parameters.AddWithValue("@p2", int.Parse(kullan.Text));
    komut3.Parameters.AddWithValue("@p3", int.Parse(bolge.Text));
    komut3.Parameters.AddWithValue("@p4", int.Parse(alan.Text));
    komut3.ExecuteNonQuery();
    MessageBox.Show("Güncelleme işlemi başarılı! ", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    baglanti.Close() ;
}
}
}
```



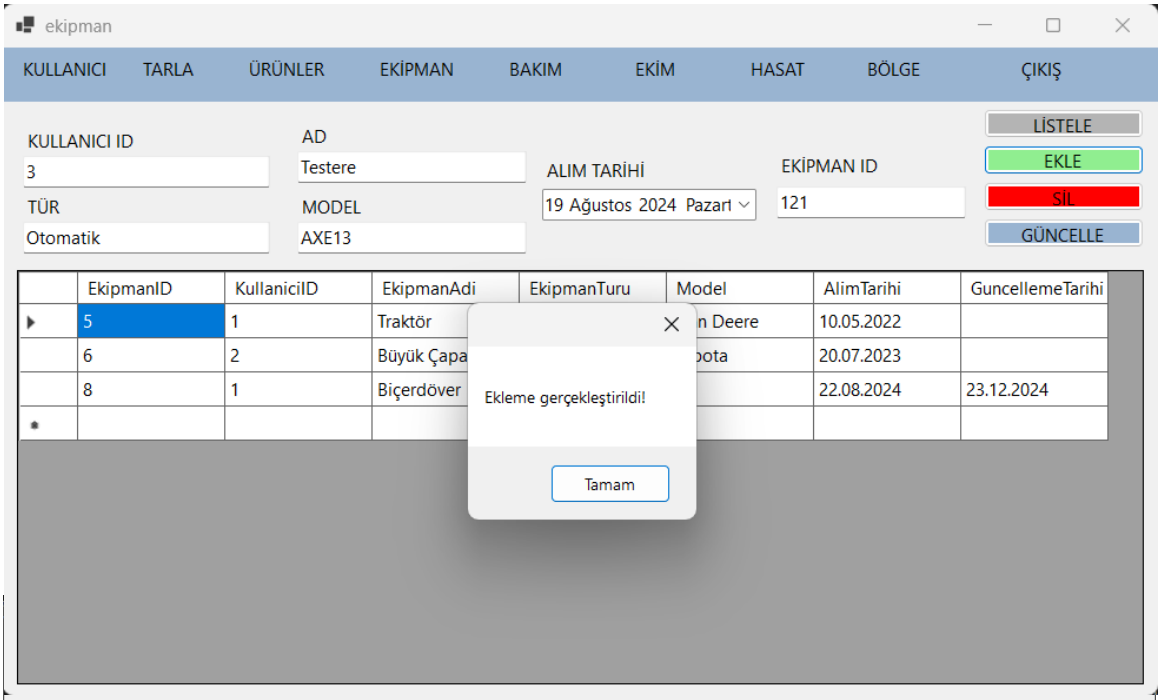
```
NpgsqlConnection baglanti = new NpgsqlConnection("server = localHost; port=5432; Database=tarimProje; user ID=postgres; password=[redacted]");
1 başvuru
private void button2_Click(object sender, EventArgs e)
{
    string sorgu = "select*from ekipman";
    NpgsqlDataAdapter da = new NpgsqlDataAdapter(sorgu, baglanti);
    DataSet ds = new DataSet();
    da.Fill(ds);
    dataGridView1.DataSource = ds.Tables[0];
}

1 başvuru
private void button1_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komut1 = new NpgsqlCommand("INSERT INTO public.\"ekipman\" (\"EkipmanID\",\"KullaniciID\",\"EkipmanAdi\",\"EkipmanTuru\",\"Model\",\"AlimTarihi\") VALUES (@p1,@p2,@p3,@p4,@p5,@p6)", baglan
    komut1.Parameters.AddWithValue("@p1", int.Parse(id.Text));
    komut1.Parameters.AddWithValue("@p2", int.Parse(kullan.Text));
    komut1.Parameters.AddWithValue("@p3", ad.Text);
    komut1.Parameters.AddWithValue("@p4", tur.Text);
    komut1.Parameters.AddWithValue("@p5", model.Text);
    komut1.Parameters.AddWithValue("@p6", dateTimePicker1.Value);
    komut1.ExecuteNonQuery();
    baglanti.Close();
    MessageBox.Show("Ekleme gerçekleştirildi!");
}

1 başvuru
private void buttonsil_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komut2 = new NpgsqlCommand("DELETE FROM public.\"ekipman\" WHERE \"EkipmanID\"=@p1", baglanti);
    komut2.Parameters.AddWithValue("@p1", int.Parse(id.Text));
    komut2.ExecuteNonQuery();
    baglanti.Close();
    MessageBox.Show("Silinme işlemi başarılı! ", "Bilgi",
        MessageBoxButtons.OK, MessageBoxIcon.Stop);
}

1 başvuru
private void btnedit_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komut3 = new NpgsqlCommand("UPDATE public.\"ekipman\" SET  \"KullaniciID\"=@p2,\"EkipmanAdi\"=@p3 ,\"EkipmanTuru\"=@p4,\"Model\"=@p5, \"AlimTarihi\"=@p6 WHERE \"EkipmanID\"=@p1", baglanti)
    komut3.Parameters.AddWithValue("@p1", int.Parse(id.Text));
    komut3.Parameters.AddWithValue("@p2", int.Parse(kullan.Text));
    komut3.Parameters.AddWithValue("@p3", ad.Text);
    komut3.Parameters.AddWithValue("@p4", tur.Text);
    komut3.Parameters.AddWithValue("@p5", model.Text);
    komut3.Parameters.AddWithValue("@p6", dateTimePicker1.Value);
    komut3.ExecuteNonQuery();
    MessageBox.Show("Güncelleme işlemi başarılı! ", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Stop);
    baglanti.Close();
}
}
}
```

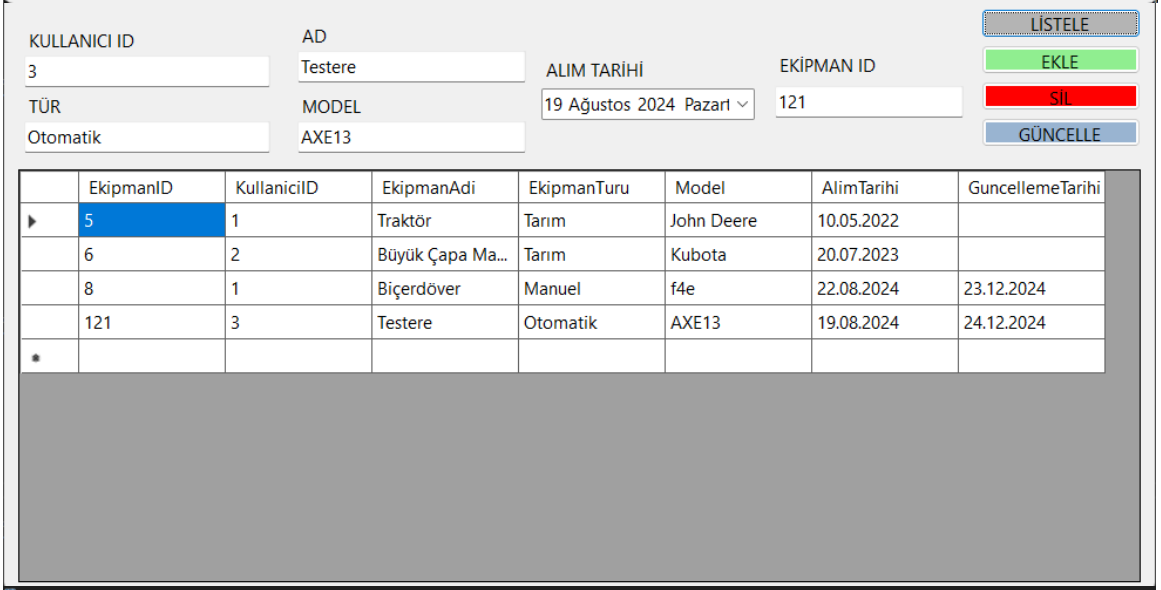
Başlangıç sayfasından “Bireysel Kullanıcı” ya da “Kurumsal Kullanıcı” olmak üzere iki seçenek sunan comboBox yardımıyla seçim yapılarak giriş yapılır ve tablolarımıza erişim sağlanır.

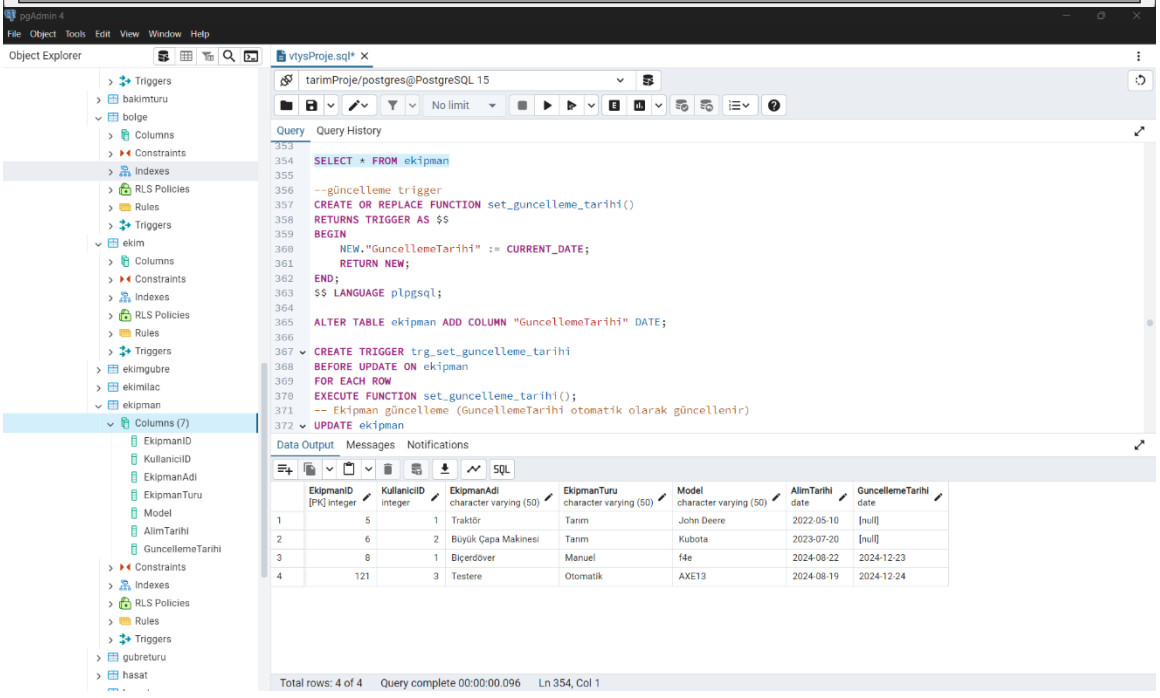




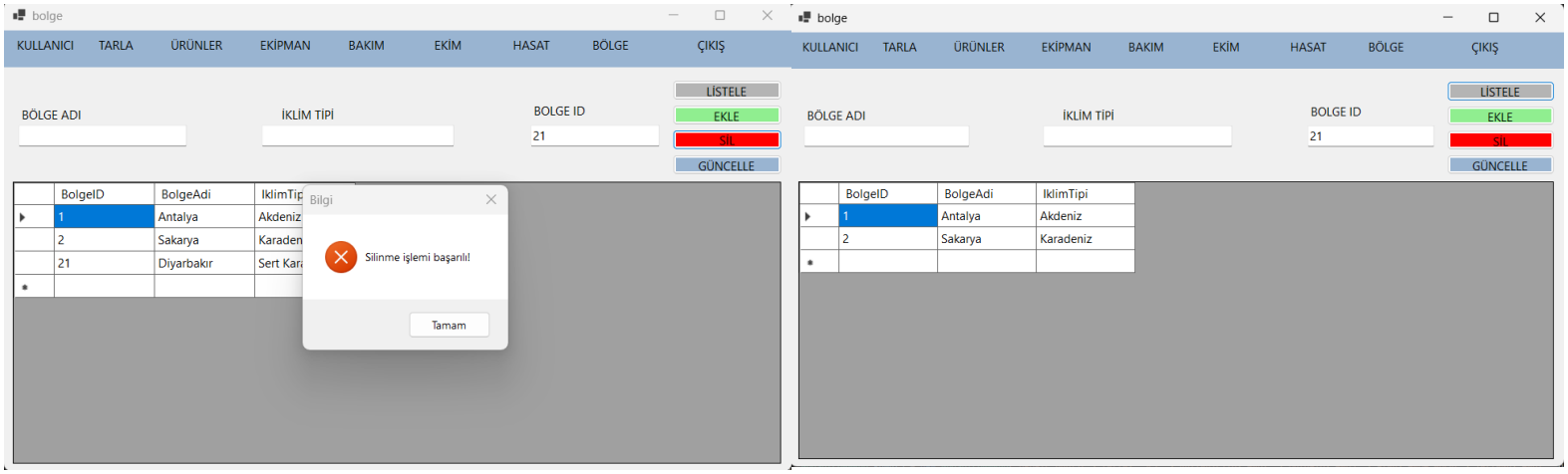
“ekipman” tablosunda ekleme örneği ve güncelleme triggerımızın işlevi olan güncelleme tarihi ekleme işleminin başarıyla gerçekleştiğine dair görüntüler:



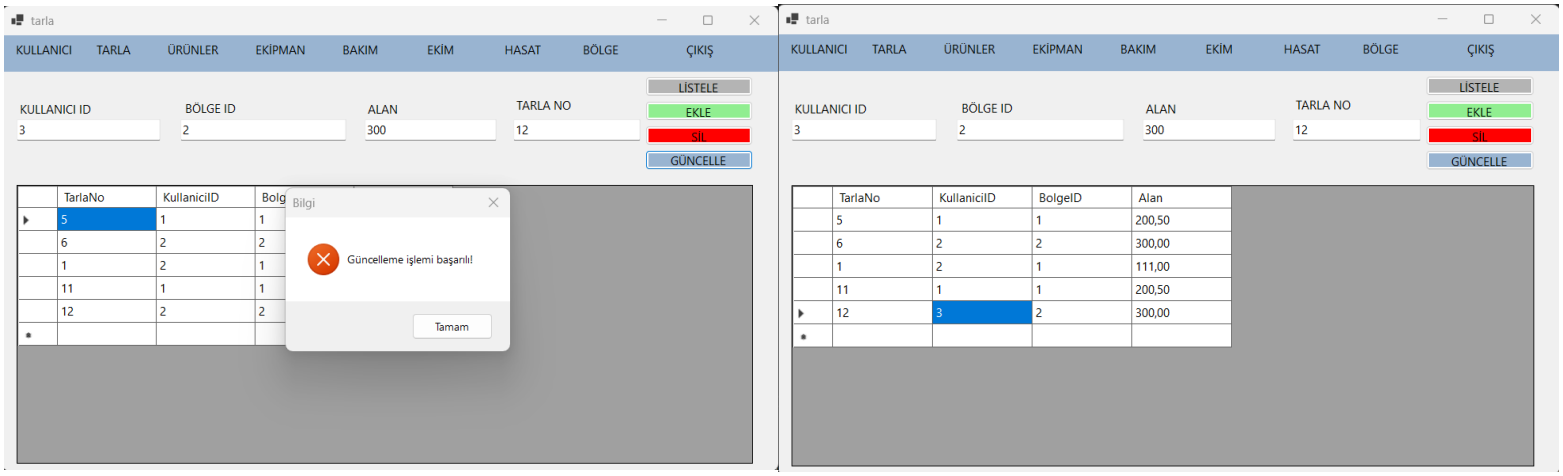




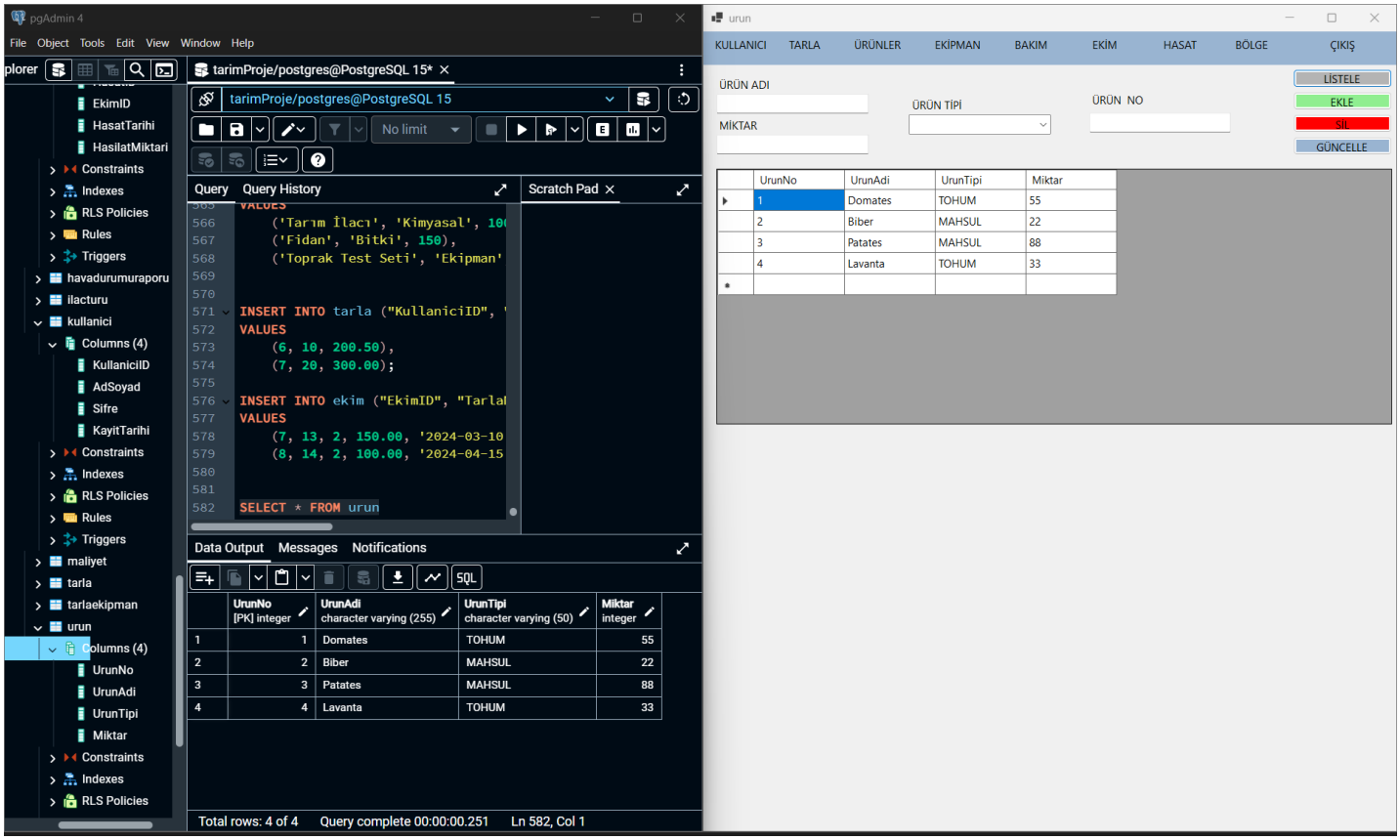
“bölge” tablomuzda listelenen tablodan bir veri satırının ID’si girilerek ilgili satırın işleme alınıp sil butonuna basıldıktan sonra başarı mesajının yansıtılması ve tekrar listelendiğinde verinin başarıyla silinip tablonun yeni halinin listelenmesi:

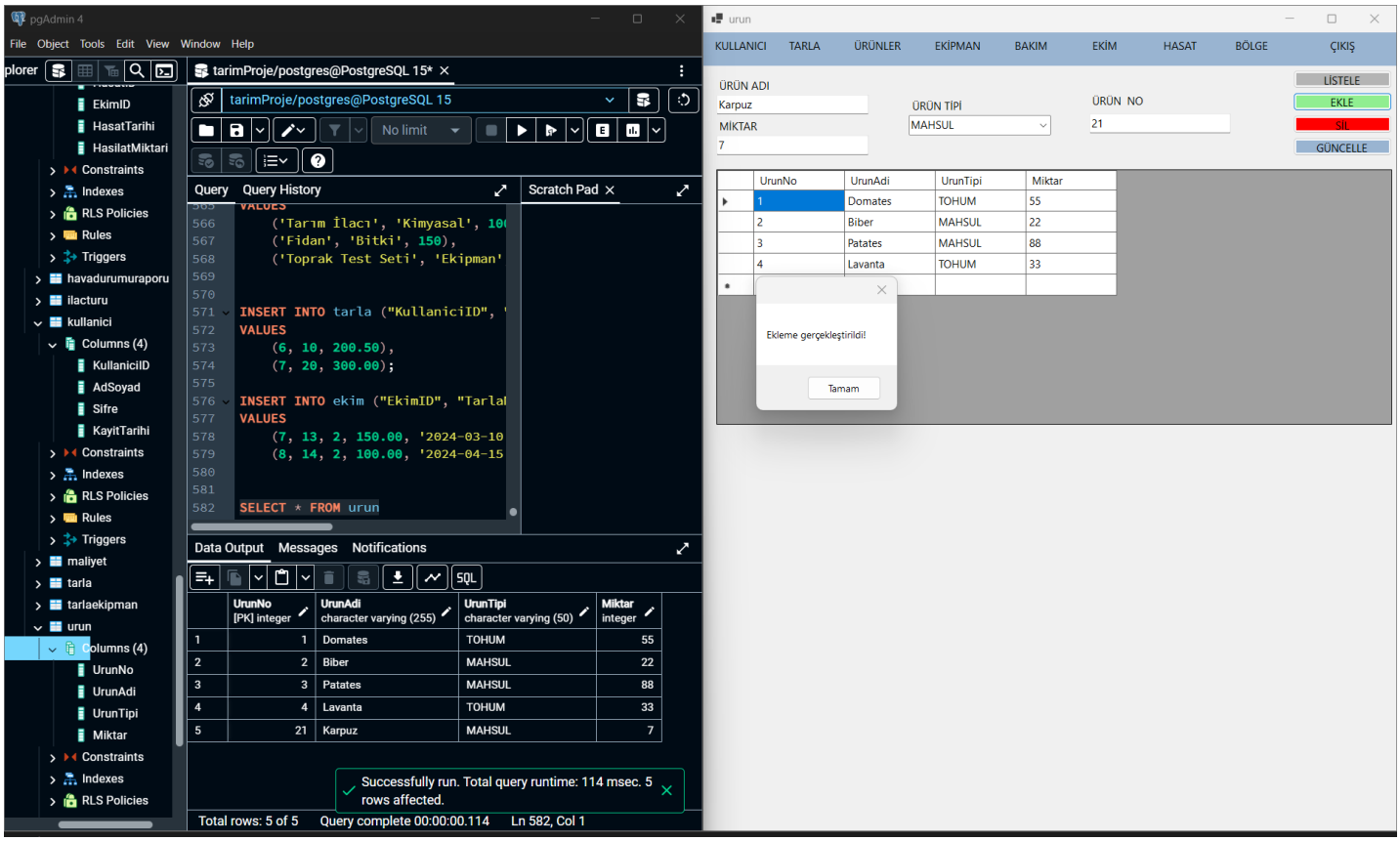


“tarla” tablosunda sırasıyla listeleme, silme ve listeleme işlemleri sonucu tablonun güncellenmesi:



Yapılan işlemlerin SQL ortamına başarıyla yansıdığının kontrolü:





Uygulama Kaynak Kodları GitHub Linki : [aystskan/VTYS_PROJE: 202425_GÜZ_VTYS_PROJE_TARIM]