SAKARYA ÜNİVERSİTESİ
BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Veri Tabanı Yönetim Sistemleri

Adı: *Mehmet*

Soyadı: *Bosdancı*

No: G221210045

E-Posta: *mehmet.bosdanci@ogr.sakarya.edu.tr*

SENARYO

Bir besi üretim yerine ait çalışanlar, hayvanlar, veteriner vb. bilgilerin tutulduğu masaüstü uygulamasıdır.

İŞ KURALLARI

Bir besi çiftliğinde inekler, buzağılar, boğalar bulunur.

İşletmedeki hayvanların ortak özelliği bulunur. (KupeNo,İsim,Yas)

İşletmede işçiler ve çalışan yakınlarına ait bilgiler bulunur.

İşçilerin sigortası bulunur.

İşletmede satış yerlerinin belirlendiği bayiler, il ve ilçe bilgileri tutulur.

Süt birliği buzağı desteği ve süt desteği sağlar.

İşletmenin aylık ürettiği süt litresi ve yıllık et üretim bilgileri bulunur.

Üretilen ürünlerin çeşitleri bulunur. (et, süt, sucuk, tereyağı, peynir).

Satış için satış fiyatı belirlenir.

İşletmedeki hayvanların sağlığı veterinerler tarafından kontrol edilir.

Hayvan sağlığında ilaç parası, tedavi ücreti, tedavi tipi bilgileri bulunur.

Belirli tipteki ihtiyaçları karşılamak için Tedarikçi bulunur. Vade imkanının olup olmaması önemlidir.

Hayvanlar yalnızca inek, buzağı ve boğadan biri olabilir.

Bir ineğin aynı anda yalnızca bir süt birliğinde bilgileri bulunurken bir süt birliğinde birden fazla inek kaydı olabilir.

Bir buzağının aynı anda yalnızca bir süt birliğinde bilgileri bulunurken bir süt birliğinde birden fazla buzağı kaydı olabilir.

Bir üretimde birden çok inek bulunurken bir inek aynı anda sadece bir üretime katkı sağlar.

Bir üretimde birden çok boğa bulunurken bir boğa aynı anda sadece bir üretime katkı sağlar.

Bir üretimin çok sayıda satışı bulunurken bir satışta çok satışta üretim olur.

Bir veteriner birden çok hayvanı tedavi edebilirken bir hayvan yalnızca bir veterinerden tedavi olur.

Bir işçi birden fazla hayvana bakabilirken bir hayvan yalnızca bir işçi tarafından bakılır.

Yalnızca bir işçinin yalnızca bir sigortası olur.

İşçilerin birden fazla çalışan yakını varken bir çalışan yakını sadece bir işçinin yakını olmalıdır.

Bir işçi birden fazla tedarikçiyle ilgilenirken bir tedarikçi yalnızca bir işçiyle irtibata geçer.

Bir işçi bir bayii açabilirken bir bayide birden fazla işçi bulunabilir.

Bir ilde birden fazla bayii bulunabilirken aynı bayii yalnızca bir ilde bulunabilir.

Bir ilçede birden fazla bayii bulunabilirken aynı bayii yalnızca bir ilçede bulunabilir.

Bir ilde birden çok ilçe bulunurken bir ilçe yalnızca bir ile aittir.

İLİŞKİSEL ŞEMA (METİNSEL GÖSTERİM)

Hayvanlar (**KupeNo**: serial, İsim: varchar, Yas: int, HayvanTipi: varchar, Veteriner: int, IsciNo:int)

Inekler (**KupeNo**: int, Sutlitresi: int, SagımSaati: time, SagımAdedi:int, SutBirligi:int, Uretim:int)

Buzagilar (**KuperNo**: int, SuttenKesimTarihi: date, GunlukIctigiSutLitresi: int, MamaIciyorMu: boolean, SutBirligi: int, Anne: int)

Bogalar (**KupeNo**: int, CanliAgirlik: int, KesimTarihi: date, DamizlikMi: boolean, Uretim: int)

SutBirligi (BirlikNo: serial, BuzagiDestegi: int, SutDestegi: int)

Uretim (UretimNo: serial, AylikSüt: int, YillikEt: int)

Kategori (KategoriID: serial, UrunTipi: varchar, UrunMaliyet: int, UretimNo: int, TezgahID:int)

Satis (TezgahID: serial, SatisFiyati: int)

SaglikVeteriner (VeterinerID: serial, VeterinerAd: varchar, VeterinerSoyaAd: varchar, TedaviTipi: varchar, TedaviUcreti: int)

Isciler (IsciNo: serial, IsciAd: varchar, IsciSoyad: varchar, KidemliMi: boolean, CalismaSaatleri: time, Maas: int, Sigorta: int, Bayii: int)

Tedarikci (TedarikciNo: serial, TedarikciTipi: varchar, VadeVarMi: boolean, Isciler: int)

CalisanYakini (YakinNo: serial, YakinIsmi: varchar, YakinTel: varchar, Isciler: int)

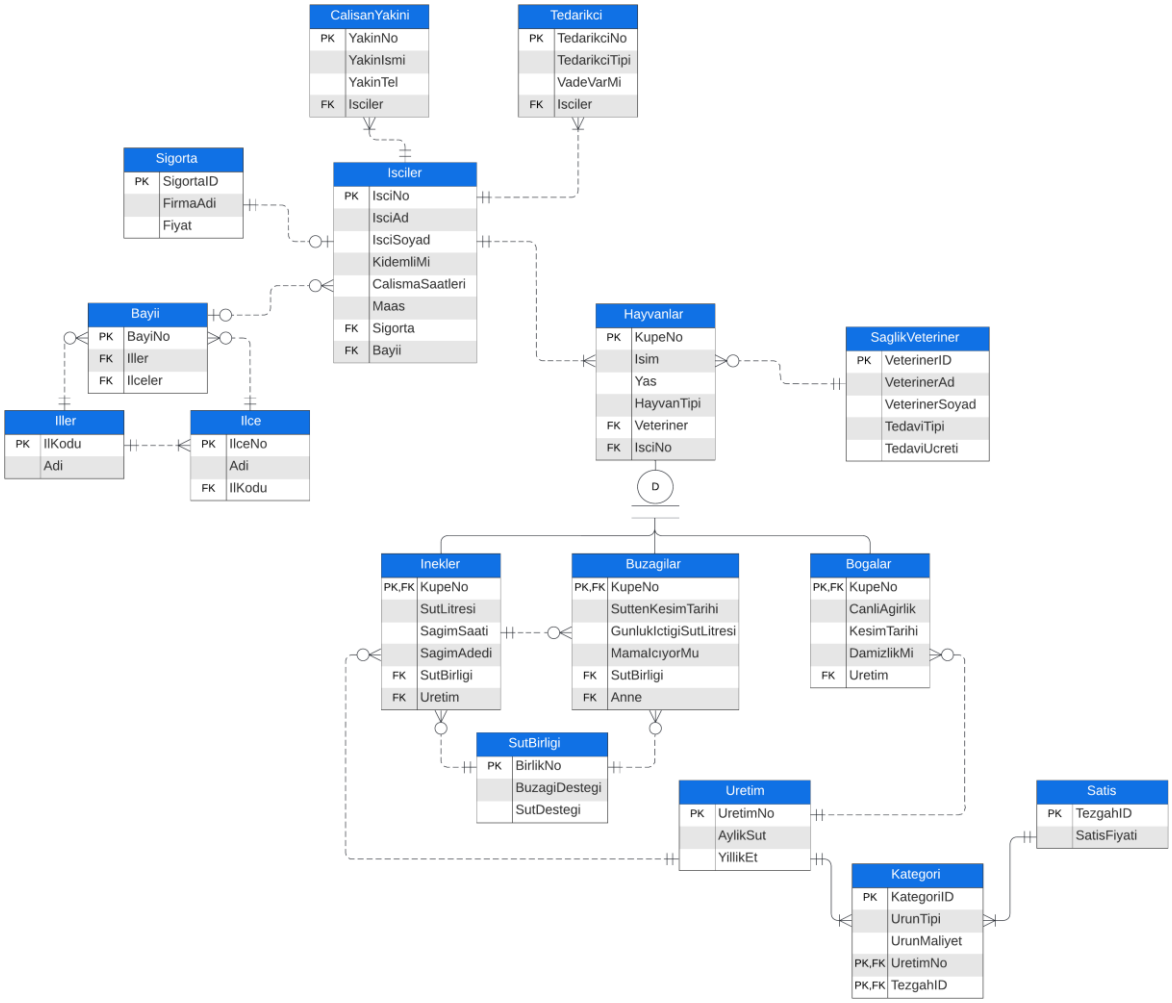
Sigorta (SigortaID: serial, FirmaAdi: varchar, Fiyat: int)

Bayii (BayiiNo: serial, Iller: int, Ilceler:int)

Iller (IlKodu: serial, Adi: varchar)

Ilceler (IlceNo: serial, Adi: varchar, IlKodu: int)

VARLIK BAĞINTI MODELİ



SQL İFADELER

```
CREATE DATABASE "BesiCiftligi";

CREATE TABLE"Hayvanlar" (
    "KupeNo" SERIAL,
    "Isim" VARCHAR(20),
    "Yas" INT,
    "HayvanTipi" VARCHAR(20),
```

```
    "Veteriner" INT,
    "IsciNo" INT,
    CONSTRAINT "HayvanlarPK" PRIMARY KEY("KupeNo")
);

CREATE TABLE "Inekler" (
    "KupeNo" INT,
    "SutLitresi" INT,
    "SagimSaati" TIME,
    "SagimAdedi" INT,
    "SutBirligi" INT,
    "Uretim" INT,
    CONSTRAINT "IneklerPK" PRIMARY KEY("KupeNo")
);
CREATE TABLE "Buzagilar" (
    "KupeNo" INT,
    "SuttenKesimTarihi" DATE,
    "GunlukIctigiSutLitresi" INT,
    "MamaIciyorMu" BOOLEAN,
    "SutBirligi" INT,
    "Anne" INT,
    CONSTRAINT "BuzagilarPK" PRIMARY KEY("KupeNo")
);
CREATE TABLE "Bogalar" (
    "KupeNo" INT,
    "CanliAgirlik" INT,
```

```
    "KesimTarihi" DATE,
    "DamizlikMi" BOOLEAN,
    "Uretim" INT,
    CONSTRAINT "BogalarPK" PRIMARY KEY("KupeNo")
);
CREATE TABLE "SutBirligi" (
    "BirlikNo" SERIAL,
    "BuzagiDestegi" INT,
    "SutDestegi" INT,
    CONSTRAINT "SutBirligiPK" PRIMARY KEY("BirlikNo")
);
CREATE TABLE "Uretim" (
    "UretimNo" SERIAL,
    "AylikSut" INT,
    "YillikEt" INT,
    CONSTRAINT "UretimPK" PRIMARY KEY("UretimNo")
);
CREATE TABLE "Kategori" (
    "KategoriID" SERIAL,
    "UrunTipi" VARCHAR(20),
    "UrunMaliyet" INT,
    "UretimNo" INT,
    "TezgahID" INT,
    CONSTRAINT "KategoriPK" PRIMARY KEY("KategoriID")
);
CREATE TABLE "Satis" (
```

```
    "TezgahID" SERIAL,
    "SatisFiyati" INT,
    CONSTRAINT "SatisPK" PRIMARY KEY("TezgahID")
);
CREATE TABLE "SaglikVeteriner" (
    "VeterinerID" SERIAL,
    "VeterinerAd" VARCHAR(20),
    "VeterinerSoyad" VARCHAR(20),
    "TedaviTipi" VARCHAR(20),
    "TedaviUcreti" INT,
    CONSTRAINT "VeterinerPK" PRIMARY KEY("VeterinerID")
);
CREATE TABLE "Isciler" (
    "IsciNo" SERIAL,
    "IsciAd" VARCHAR(20),
    "IsciSoyad" VARCHAR(20),
    "KidemliMi" BOOLEAN,
    "CalismaSaatleri" TIME,
    "Maas" INT,
    "Sigorta" INT,
    "Bayii" INT,
    CONSTRAINT "IscilerPK" PRIMARY KEY("IsciNo")
);
CREATE TABLE "Tedarikci" (
    "TedarikciNo" SERIAL,
    "TedarikciTipi" VARCHAR(20),
```

```
    "VadeVarMi" BOOLEAN,
    "Isciler" INT,
    CONSTRAINT "TedarikciPK" PRIMARY KEY("TedarikciNo")
);
CREATE TABLE "CalisanYakini" (
    "YakinNo" SERIAL,
    "YakinIsmi" VARCHAR(20),
    "YakinTel" VARCHAR(20),
    "Isciler" INT,
    CONSTRAINT "YakinPK" PRIMARY KEY("YakinNo")
);
CREATE TABLE "Sigorta" (
    "SigortaID" SERIAL,
    "FirmaAdi" VARCHAR(20),
    "Fiyat" INT,
    CONSTRAINT "SigortaPK" PRIMARY KEY("SigortaID")
);
CREATE TABLE "Bayii" (
    "BayiNo" SERIAL,
    "Iller" INT,
    "Ilceler" INT,
    CONSTRAINT "BayiiPK" PRIMARY KEY("BayiNo")
);
CREATE TABLE "Iller" (
    "IlKodu" SERIAL,
    "Adi" VARCHAR(20),
```

```
   CONSTRAINT "IllerPK" PRIMARY KEY("IlKodu")
);
CREATE TABLE "Ilce" (
   "IlceNo" SERIAL,
   "Adi" VARCHAR(20),
   "IlKodu" INT,
   CONSTRAINT "IlcePK" PRIMARY KEY("IlceNo")
);
ALTER TABLE "Hayvanlar"
        ADD CONSTRAINT "VeterinerFK" FOREIGN KEY ("Veteriner")
        REFERENCES "SaglikVeteriner" ("VeterinerID")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Hayvanlar"
   ADD CONSTRAINT "IsciNoFK" FOREIGN KEY ("IsciNo")
        REFERENCES "Isciler" ("IsciNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Inekler"
        ADD CONSTRAINT "IneklerFK" FOREIGN KEY ("KupeNo")
        REFERENCES "Hayvanlar" ("KupeNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;
```

```
ALTER TABLE "Inekler"
        ADD CONSTRAINT "SutBirligiFK" FOREIGN KEY ("SutBirligi")
        REFERENCES "SutBirligi" ("BirlikNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Inekler"
        ADD CONSTRAINT "UretimFK" FOREIGN KEY ("Uretim")
        REFERENCES "Uretim" ("UretimNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Buzagilar"
        ADD CONSTRAINT "BuzagiFK" FOREIGN KEY ("KupeNo")
        REFERENCES "Hayvanlar" ("KupeNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Buzagilar"
        ADD CONSTRAINT "SutBirligiFK" FOREIGN KEY ("SutBirligi")
        REFERENCES "SutBirligi" ("BirlikNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Buzagilar"
        ADD CONSTRAINT "AnneFK" FOREIGN KEY ("Anne")
```

```
        REFERENCES "Inekler" ("KupeNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Bogalar"
        ADD CONSTRAINT "BogalarFK" FOREIGN KEY ("KupeNo")
        REFERENCES "Hayvanlar" ("KupeNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Bogalar"
        ADD CONSTRAINT "UretimFK" FOREIGN KEY ("Uretim")
        REFERENCES "Uretim" ("UretimNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Kategori"
        ADD CONSTRAINT "UretimNoFK" FOREIGN KEY ("UretimNo")
        REFERENCES "Uretim" ("UretimNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Kategori"
        ADD CONSTRAINT "TezgahIDFK" FOREIGN KEY ("TezgahID")
        REFERENCES "Satis" ("TezgahID")
        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE;


ALTER TABLE "Isciler"
        ADD CONSTRAINT "SigortaFK" FOREIGN KEY ("Sigorta")
        REFERENCES "Sigorta" ("SigortaID")
        ON DELETE CASCADE
        ON UPDATE CASCADE;


ALTER TABLE "Isciler"
        ADD CONSTRAINT "BayiiFK" FOREIGN KEY ("Bayii")
        REFERENCES "Bayii" ("BayiNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;


ALTER TABLE "Tedarikci"
        ADD CONSTRAINT "IscilerFK" FOREIGN KEY ("Isciler")
        REFERENCES "Isciler" ("IsciNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;


ALTER TABLE "CalisanYakini"
        ADD CONSTRAINT "IscilerFK" FOREIGN KEY ("Isciler")
        REFERENCES "Isciler" ("IsciNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;
```

```
ALTER TABLE "Bayii"
        ADD CONSTRAINT "IllerFK" FOREIGN KEY ("Iller")
        REFERENCES "Iller" ("IlKodu")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Bayii"
        ADD CONSTRAINT "IlcelerFK" FOREIGN KEY ("Ilceler")
        REFERENCES "Ilce" ("IlceNo")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

ALTER TABLE "Ilce"
        ADD CONSTRAINT "IlKoduFK" FOREIGN KEY ("IlKodu")
        REFERENCES "Iller" ("IlKodu")
        ON DELETE CASCADE
        ON UPDATE CASCADE;

INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi")
VALUES ('Akkiz','4','Inek')

INSERT INTO "SaglikVeteriner"("VeterinerAd","VeterinerSoyad","TedaviTipi","TedaviUcreti")
VALUES ('Ahmet','Akbalci','Tirnak Bakimi', '150')

INSERT INTO "SaglikVeteriner"("VeterinerAd","VeterinerSoyad","TedaviTipi","TedaviUcreti")
VALUES ('Mehmet','BagriAcik','Dis Parazit', '100')
```

```
INSERT INTO "SaglikVeteriner"("VeterinerAd","VeterinerSoyad","TedaviTipi","TedaviUcreti")
VALUES ('Orhan','GunDogdu','Dis Parazit', '120')

INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi","Veteriner")
VALUES ('Duman','1','Buzagi', '1')

INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi","Veteriner")
VALUES ('Cuma','1','Buzagi', '2')

INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi","Veteriner")
VALUES ('Ramazan','3','Boga', '3')

INSERT INTO "Isciler"("IsciAd","IsciSoyad","KidemliMi","Maas")
VALUES ('Mustafa','Yildiz',TRUE, '21000')

INSERT INTO "Sigorta"("FirmaAdi","Fiyat")
VALUES ('SaglamSigorta','7000')

INSERT INTO "Sigorta"("FirmaAdi","Fiyat")
VALUES ('GuvenSigorta','8000')

INSERT INTO "Iller"("IlKodu","Adi")
VALUES ('42','Konya')

INSERT INTO "Ilce"("IlceNo","Adi")
```

```
VALUES ('1','Bozkir')

INSERT INTO "Bayii"("Iller","Ilceler")
VALUES ('42','1')

INSERT INTO "Isciler"("IsciAd","IsciSoyad","KidemliMi","Maas", "Sigorta")
VALUES ('Hamit','Aslan',FALSE, '18000','1')

INSERT INTO "Isciler"("IsciAd","IsciSoyad","KidemliMi","Maas", "Sigorta","Bayii")
VALUES ('Yusuf','AltinBas',TRUE, '22000','2','3')

INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi","Veteriner","IsciNo")
VALUES ('Toprak','2','Boga', '2','1')

INSERT INTO "Inekler"("KupeNo","SutLitresi", "SagimAdedi")
VALUES ('1','25','2');

INSERT INTO "Inekler"("KupeNo","SutLitresi", "SagimAdedi","SagimSaati")
VALUES ('6','20','1','08:00');

INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi","Veteriner")
VALUES ('Nur','3','Inek', '1')

INSERT INTO "SutBirligi"("BuzagiDestegi","SutDestegi")
VALUES ('500','2')
```

```
INSERT INTO "SutBirligi"("BuzagiDestegi","SutDestegi")

VALUES ('750','5')


INSERT INTO
"Buzagilar"("KupeNo","SuttenKesimTarihi","GunlukIctigiSutLitresi","Anne","SutBirligi")

VALUES ('4','21.02.2023','5', '1','1')


INSERT INTO
"Buzagilar"("KupeNo","SuttenKesimTarihi","GunlukIctigiSutLitresi","Anne","SutBirligi","Mam
aIciyorMu")

VALUES ('5','14.03.2023','6', '6','2',TRUE)


INSERT INTO "Uretim"("YillikEt")

VALUES ('950')


INSERT INTO "Uretim"("YillikEt")

VALUES ('1000')


INSERT INTO "Uretim"("AylikSut")

VALUES ('600')


INSERT INTO "Uretim"("AylikSut")

VALUES ('800')


INSERT INTO "Bogalar"("KupeNo","CanliAgirlik","KesimTarihi","DamizlikMi","Uretim")

VALUES ('7','800','23.11.2023', FALSE,'1')
```

```
INSERT INTO "Bogalar"("KupeNo","CanliAgirlik","KesimTarihi","DamizlikMi","Uretim")

VALUES ('8','700',NULL, TRUE,'2')


INSERT INTO "Hayvanlar"("Isim","Yas","HayvanTipi","Veteriner","IsciNo")

VALUES ('Akasya','5','Inek', '3','3')


INSERT INTO "Inekler"("KupeNo","SutLitresi",
"SagimAdedi","SagimSaati","Uretim","SutBirligi")

VALUES ('9','28','1','08:05','3','2');


INSERT INTO "CalisanYakini"("YakinIsmi","YakinTel","Isciler")

VALUES ('Furkan','0555 444 22 55','1')


INSERT INTO "CalisanYakini"("YakinIsmi","YakinTel","Isciler")

VALUES ('Halit','0333 111 22 78','2')


INSERT INTO "CalisanYakini"("YakinIsmi","YakinTel","Isciler")

VALUES ('Kazim','0666 441 36 53','2')


INSERT INTO "CalisanYakini"("YakinIsmi","YakinTel","Isciler")

VALUES ('Ali','0777 887 43 21','3')

-----------------------------------------------------------------------------------------

--FONKSİYONLARR

-----------------------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION getVeterinerInfo()

RETURNS TABLE(VeterinerID INT, VeterinerAd VARCHAR(20), VeterinerSoyad
VARCHAR(20),TedaviTipi VARCHAR(20),TedaviUcreti INT) AS $$
```

```
BEGIN
    RETURN QUERY
    SELECT "VeterinerID", "VeterinerAd", "VeterinerSoyad", "TedaviTipi", "TedaviUcreti"
FROM "SaglikVeteriner";
END;
$$ LANGUAGE plpgsql;

SELECT * FROM getVeterinerInfo();
-----------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION getCalisanYakini()
RETURNS TABLE(YakinNo INT, YakinIsmi VARCHAR(20), YakinTel VARCHAR(20), Isciler INT) AS
$$
BEGIN
    RETURN QUERY
    SELECT "YakinNo", "YakinIsmi", "YakinTel", "Isciler" FROM "CalisanYakini";
END;
$$ LANGUAGE plpgsql;

SELECT * FROM getCalisanYakini();
-----------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION getInekler()
RETURNS TABLE(KupeNo INT, Isim VARCHAR(20), Yas INT, Veteriner INT) AS $$
BEGIN
    RETURN QUERY
    SELECT "KupeNo", "Isim", "Yas", "Veteriner"
    FROM "Hayvanlar"
    WHERE "HayvanTipi" = 'Inek';
```

```
END;
$$ LANGUAGE plpgsql;


SELECT * FROM getInekler();
---------------------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION getIsciler()
RETURNS TABLE(IsciNo INT, IsciAd VARCHAR(20), IsciSoyad VARCHAR(20), KidemliMi
BOOLEAN, Maas INT, Sigorta INT) AS $$
BEGIN
   RETURN QUERY
   SELECT "IsciNo", "IsciAd", "IsciSoyad", "KidemliMi", "Maas", "Sigorta"
   FROM "Isciler";
END;
$$ LANGUAGE plpgsql;


SELECT * FROM getIsciler();
---------------------------------------------------------------------------------------------
--ORTALAMA MAAS FONKSİYONU
CREATE OR REPLACE FUNCTION OrtalamaMaas()
RETURNS DECIMAL AS $$
DECLARE
   ortalama_maas DECIMAL;
BEGIN
   SELECT AVG("Maas") INTO ortalama_maas FROM "Isciler";
   RETURN ortalama_maas;
END;
```

```
$$ LANGUAGE PLPGSQL;


SELECT OrtalamaMaas();

--
**************************************************************************
**********************

--işçi maası güncelleme

CREATE OR REPLACE FUNCTION IsciMaasiGucelleme(employee_id INT, new_salary INT)

RETURNS VOID AS $$

BEGIN

   UPDATE "Isciler"

   SET "Maas" = new_salary

   WHERE "IsciNo" = employee_id;

END;

$$ LANGUAGE plpgsql;


SELECT IsciMaasiGucelleme(1, 21000);--işci maas güncelleme

--*****************************************************************

--trigger

--*****************************************************************

-- IsciTrigger adında bir tablo oluşturuyorum

CREATE TABLE IsciTrigger (

   "IsciNo" SERIAL,

   "Ad" VARCHAR(20),

   "Soyad" VARCHAR(20),

   "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   "EskiMaas" INT,
```

```
    "YeniMaas" INT
);

-- Tetikleyici (Trigger) nin devreye girmesi
CREATE OR REPLACE FUNCTION IsciMaasDegisikligi()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW."Maas" <> OLD."Maas" THEN
        INSERT INTO IsciTrigger ("IsciNo", "Ad", "Soyad", "EskiMaas", "YeniMaas")
        VALUES (NEW."IsciNo", NEW."IsciAd", NEW."IsciSoyad", OLD."Maas", NEW."Maas");
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Tetikleyici'nin Isciler tablosunda tetiklenmesi
CREATE TRIGGER salaryChangeTrigger
AFTER UPDATE OF "Maas" ON "Isciler"
FOR EACH ROW EXECUTE FUNCTION IsciMaasDegisikligi();

DELETE FROM IsciTrigger--IsciTrigger tablosundaki verileri temizleme
-------------------------------------------------------------------------------
CREATE OR REPLACE FUNCTION VeterinerUcretDegistirme(vet_id INT, yeni_ucret INT)
RETURNS VOID AS $$
BEGIN
    UPDATE "SaglikVeteriner"
```

```
    SET "TedaviUcreti" = yeni_ucret
    WHERE "VeterinerID" = vet_id;
END;
$$ LANGUAGE plpgsql;

SELECT VeterinerUcretDegistirme(1,150);--ücret değiştirme
-------------------------------------------------------------------------------
--Veteriner triggeri
CREATE TABLE IF NOT EXISTS VeterinerTrigger (
    "VeterinerID" SERIAL,
    "VeterinerAd" VARCHAR(20),
    "VeterinerSoyad" VARCHAR(20),
    "TedaviTipi" VARCHAR(20),
    "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "EskiTedaviUcreti" INT,
    "YeniTedaviUcreti" INT
);

CREATE OR REPLACE FUNCTION TedaviUcretiDegisimTetikleme()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW."TedaviUcreti" <> OLD."TedaviUcreti" THEN
        INSERT INTO VeterinerTrigger ("VeterinerID", "VeterinerAd", "VeterinerSoyad",
"TedaviTipi", "EskiTedaviUcreti", "YeniTedaviUcreti")
        VALUES (NEW."VeterinerID", NEW."VeterinerAd", NEW."VeterinerSoyad",
NEW."TedaviTipi", OLD."TedaviUcreti", NEW."TedaviUcreti");
    END IF;
```

```
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER TedaviUcretiDegisim
AFTER UPDATE OF "TedaviUcreti" ON "SaglikVeteriner"
FOR EACH ROW EXECUTE FUNCTION TedaviUcretiDegisimTetikleme();

DELETE FROM VeterinerTrigger--VeterinerTrigger tablosundaki verileri temizleme
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
--Süt birliğindeki Buzagi destegini degiştirme fonksiyonu
CREATE OR REPLACE FUNCTION BuzagiDestegiGuncelle(birlik_id INT, yeni_deger INT)
RETURNS VOID AS $$
BEGIN
    UPDATE "SutBirligi"
    SET "BuzagiDestegi" = yeni_deger
    WHERE "BirlikNo" = birlik_id;
END;
$$ LANGUAGE plpgsql;

SELECT BuzagiDestegiGuncelle(1, 500); -- birlikNo suna göre buzağı destegi değiştirme

-----------------------------------------------------------------------------------------------
--BuzagiDestegiTrigger
CREATE TABLE IF NOT EXISTS BuzagiDestegiTrigger (
```

```
    "BirlikNo" SERIAL,
    "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    "EskiBuzagiDestegi" INT,
    "YeniBuzagiDestegi" INT
);

CREATE OR REPLACE FUNCTION BuzagiDestegiDegisimTetikleme()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW."BuzagiDestegi" <> OLD."BuzagiDestegi" THEN
        INSERT INTO BuzagiDestegiTrigger ("BirlikNo", "EskiBuzagiDestegi", "YeniBuzagiDestegi")
        VALUES (NEW."BirlikNo", OLD."BuzagiDestegi", NEW."BuzagiDestegi");
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER BuzagiDestegiDegisim
AFTER UPDATE OF "BuzagiDestegi" ON "SutBirligi"
FOR EACH ROW EXECUTE FUNCTION BuzagiDestegiDegisimTetikleme();

DELETE FROM BuzagiDestegiTrigger--BuzagiDestegiTrigger tablosundaki verileri temizleme
-------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------------
--Süt birliğindeki Süt destegini degiştirme fonksiyonu
```

```
CREATE OR REPLACE FUNCTION SutDestegiGuncelle(birlik_id INT, yeni_deger INT)
RETURNS VOID AS $$
BEGIN
   UPDATE "SutBirligi"
   SET "SutDestegi" = yeni_deger
   WHERE "BirlikNo" = birlik_id;
END;
$$ LANGUAGE plpgsql;


SELECT SutDestegiGuncelle(1,2); -- SutBirligi tablosundaki SutDestegi alanını BirlikNo yu kullanarak değiştirir
```

```
--SutDestegiTrigger
CREATE TABLE IF NOT EXISTS SutDestegiTrigger (
   "BirlikNo" SERIAL,
   "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   "EskiSutDestegi" INT,
   "YeniSutDestegi" INT
);
CREATE OR REPLACE FUNCTION SutDestegiDegisimTetikleme()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW."SutDestegi" <> OLD."SutDestegi" THEN
      INSERT INTO SutDestegiTrigger ("BirlikNo", "EskiSutDestegi", "YeniSutDestegi")
      VALUES (NEW."BirlikNo", OLD."SutDestegi", NEW."SutDestegi");
```

```
    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER SutDestegiDegisim

AFTER UPDATE OF "SutDestegi" ON "SutBirligi"

FOR EACH ROW EXECUTE FUNCTION SutDestegiDegisimTetikleme();


DELETE FROM SutDestegiTrigger--BuzagiDestegiTrigger tablosundaki verileri temizleme
```

FONKSİYONLAR,

```
CREATE OR REPLACE FUNCTION getVeterinerInfo()

RETURNS TABLE(VeterinerID INT, VeterinerAd VARCHAR(20), VeterinerSoyad
VARCHAR(20),TedaviTipi VARCHAR(20),TedaviUcreti INT) AS $$

BEGIN

    RETURN QUERY

    SELECT "VeterinerID", "VeterinerAd", "VeterinerSoyad", "TedaviTipi", "TedaviUcreti"
FROM "SaglikVeteriner";

END;

$$ LANGUAGE plpgsql;


SELECT * FROM getVeterinerInfo();
```

Veteriner tablosuna ait bilgilerin getirilmesini sağlar

```
CREATE OR REPLACE FUNCTION getCalisanYakini()

RETURNS TABLE(YakinNo INT, YakinIsmi VARCHAR(20), YakinTel VARCHAR(20), Isciler INT) AS
$$

BEGIN

   RETURN QUERY

   SELECT "YakinNo", "YakinIsmi", "YakinTel", "Isciler" FROM "CalisanYakini";

END;

$$ LANGUAGE plpgsql;


SELECT * FROM getCalisanYakini();
```

Çalışan yakınına ait bilgilerin getirilmesini sağlar.

```
CREATE OR REPLACE FUNCTION getInekler()

RETURNS TABLE(KupeNo INT, Isim VARCHAR(20), Yas INT, Veteriner INT) AS $$

BEGIN

   RETURN QUERY

   SELECT "KupeNo", "Isim", "Yas", "Veteriner"

   FROM "Hayvanlar"

   WHERE "HayvanTipi" = 'Inek';

END;

$$ LANGUAGE plpgsql;


SELECT * FROM getInekler();
```

İnekler tablosuna ait bilgilerin getirilmesini sağlar.

```
CREATE OR REPLACE FUNCTION getIsciler()

RETURNS TABLE(IsciNo INT, IsciAd VARCHAR(20), IsciSoyad VARCHAR(20), KidemliMi
BOOLEAN, Maas INT, Sigorta INT) AS $$

BEGIN
```

```
    RETURN QUERY
    SELECT "IsciNo", "IsciAd", "IsciSoyad", "KidemliMi", "Maas", "Sigorta"
    FROM "Isciler";
END;
$$ LANGUAGE plpgsql;

SELECT * FROM getIsciler();
```

İşçiler tablosuna ait bilgilerin getirilmesini sağlar.

```
CREATE OR REPLACE FUNCTION OrtalamaMaas()
RETURNS DECIMAL AS $$
DECLARE
    ortalama_maas DECIMAL;
BEGIN
    SELECT AVG("Maas") INTO ortalama_maas FROM "Isciler";
    RETURN ortalama_maas;
END;
$$ LANGUAGE PLPGSQL;

SELECT OrtalamaMaas();
```

İşçiler tablosundaki çalışan kişilerin ortalama maaşını getirir.

```
CREATE OR REPLACE FUNCTION IsciMaasiGucelleme(employee_id INT, new_salary INT)
RETURNS VOID AS $$
BEGIN
    UPDATE "Isciler"
    SET "Maas" = new_salary
    WHERE "IsciNo" = employee_id;
```

```
END;
$$ LANGUAGE plpgsql;

SELECT IsciMaasiGucelleme(1, 21000);
```

İşçi maaşını günceller.

```
CREATE OR REPLACE FUNCTION VeterinerUcretDegistirme(vet_id INT, yeni_ucret INT)
RETURNS VOID AS $$
BEGIN
   UPDATE "SaglikVeteriner"
   SET "TedaviUcreti" = yeni_ucret
   WHERE "VeterinerID" = vet_id;
END;
$$ LANGUAGE plpgsql;

SELECT VeterinerUcretDegistirme(1,150);
```

Veterinerlerin tedavi ücretini değiştirir.

```
CREATE OR REPLACE FUNCTION BuzagiDestegiGuncelle(birlik_id INT, yeni_deger INT)
RETURNS VOID AS $$
BEGIN
   UPDATE "SutBirligi"
   SET "BuzagiDestegi" = yeni_deger
   WHERE "BirlikNo" = birlik_id;
END;
$$ LANGUAGE plpgsql;

SELECT BuzagiDestegiGuncelle(1, 500);
```

BirlikNo suna göre buzağı destegini değiştirir.

```
CREATE OR REPLACE FUNCTION SutDestegiGuncelle(birlik_id INT, yeni_deger INT)
RETURNS VOID AS $$
BEGIN
   UPDATE "SutBirligi"
   SET "SutDestegi" = yeni_deger
   WHERE "BirlikNo" = birlik_id;
END;
$$ LANGUAGE plpgsql;


SELECT SutDestegiGuncelle(1,2);
```

SutBirligi tablosundaki SutDestegi alanını BirlikNo yu kullanarak değiştirir

---

TRİGGERLAR

```
CREATE TABLE IsciTrigger (
   "IsciNo" SERIAL,
   "Ad" VARCHAR(20),
   "Soyad" VARCHAR(20),
   "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   "EskiMaas" INT,
   "YeniMaas" INT
);


-- Tetikleyici (Trigger) nin devreye girmesi
CREATE OR REPLACE FUNCTION IsciMaasDegisikligi()
RETURNS TRIGGER AS $$
```

```
BEGIN
   IF NEW."Maas" <> OLD."Maas" THEN
      INSERT INTO IsciTrigger ("IsciNo", "Ad", "Soyad", "EskiMaas", "YeniMaas")
      VALUES (NEW."IsciNo", NEW."IsciAd", NEW."IsciSoyad", OLD."Maas", NEW."Maas");
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


-- Tetikleyici'nin Isciler tablosunda tetiklenmesi
CREATE TRIGGER salaryChangeTrigger
AFTER UPDATE OF "Maas" ON "Isciler"
FOR EACH ROW EXECUTE FUNCTION IsciMaasDegisikligi();
```

Eğer işçi tablosundaki maaş kısmında bir değişiklik olursa tetiklensin ve iscitrigger adında tablo oluşturarak IsciNo Ad Soyad DegistirilmeZamani EskiMaas YeniMaas bilgilerini kaydeder

---

```
CREATE TABLE IF NOT EXISTS VeterinerTrigger (
   "VeterinerID" SERIAL,
   "VeterinerAd" VARCHAR(20),
   "VeterinerSoyad" VARCHAR(20),
   "TedaviTipi" VARCHAR(20),
   "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   "EskiTedaviUcreti" INT,
   "YeniTedaviUcreti" INT
);


CREATE OR REPLACE FUNCTION TedaviUcretiDegisimTetikleme()
```

```
RETURNS TRIGGER AS $$

BEGIN

   IF NEW."TedaviUcreti" <> OLD."TedaviUcreti" THEN

      INSERT INTO VeterinerTrigger ("VeterinerID", "VeterinerAd", "VeterinerSoyad",
"TedaviTipi", "EskiTedaviUcreti", "YeniTedaviUcreti")

      VALUES (NEW."VeterinerID", NEW."VeterinerAd", NEW."VeterinerSoyad",
NEW."TedaviTipi", OLD."TedaviUcreti", NEW."TedaviUcreti");

   END IF;

   RETURN NEW;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER TedaviUcretiDegisim

AFTER UPDATE OF "TedaviUcreti" ON "SaglikVeteriner"

FOR EACH ROW EXECUTE FUNCTION TedaviUcretiDegisimTetikleme();
```

SaglikVeteriner tablosundaki tedavi ücreti kısmı değişirse devreye girer ve verterinertrigger isminde tablo oluşturur tabloya VeterinerID VeterinerAd VeterinerSoyad TedaviTipi DegistirilmeZamani EskiTedaviUcreti ve YeniTedaviUcreti bilgilerini kaydeder.

```
CREATE TABLE IF NOT EXISTS BuzagiDestegiTrigger (

   "BirlikNo" SERIAL,

   "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

   "EskiBuzagiDestegi" INT,

   "YeniBuzagiDestegi" INT

);


CREATE OR REPLACE FUNCTION BuzagiDestegiDegisimTetikleme()

RETURNS TRIGGER AS $$
```

```
BEGIN
   IF NEW."BuzagiDestegi" <> OLD."BuzagiDestegi" THEN
      INSERT INTO BuzagiDestegiTrigger ("BirlikNo", "EskiBuzagiDestegi", "YeniBuzagiDestegi")
      VALUES (NEW."BirlikNo", OLD."BuzagiDestegi", NEW."BuzagiDestegi");
   END IF;
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER BuzagiDestegiDegisim
AFTER UPDATE OF "BuzagiDestegi" ON "SutBirligi"
FOR EACH ROW EXECUTE FUNCTION BuzagiDestegiDegisimTetikleme();
```

SutBirligi tablosundaki BuzagiDestegi değiştirilirse tetiklenerek buzagidestegitrigger isminde bir tablo oluşturur ve tabloya BirlikNo DegistirilmeZamani EskiBuzagiDestegi YeniBuzagiDestegi bilgilerini kaydeder.

```
CREATE TABLE IF NOT EXISTS SutDestegiTrigger (
   "BirlikNo" SERIAL,
   "DegistirilmeZamani" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   "EskiSutDestegi" INT,
   "YeniSutDestegi" INT
);
CREATE OR REPLACE FUNCTION SutDestegiDegisimTetikleme()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW."SutDestegi" <> OLD."SutDestegi" THEN
      INSERT INTO SutDestegiTrigger ("BirlikNo", "EskiSutDestegi", "YeniSutDestegi")
      VALUES (NEW."BirlikNo", OLD."SutDestegi", NEW."SutDestegi");
```

```
    END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER SutDestegiDegisim

AFTER UPDATE OF "SutDestegi" ON "SutBirligi"

FOR EACH ROW EXECUTE FUNCTION SutDestegiDegisimTetikleme();
```

Eğer SutBirliği tablosundaki SutDestegi kısmı değişikliğe uğrarsa tetiklenir ve sutdestegitrigger isminde tablo oluşturarak BirlikNo DegistirilmeZamani EskiSutDestegi YeniSutDestegi bilgilerini kaydeder.

UYGULAMA KAYNAK KODLARI

Form1.cs:

```
using Npgsql;
using System.Data;

namespace VeriTabanı
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
        NpgsqlConnection baglanti = new
NpgsqlConnection("Server=localhost;Port=5432;Database=BesiCiftligi;User
Id=postgres;Password=1234;");
        private void BtnListele_Click(object sender, EventArgs e)
        {
            try
            {
                baglanti.Open();
```

```
                string sorgu = "SELECT * FROM \"SaglikVeteriner\" ORDER BY
\"VeterinerID\" ASC";
                NpgsqlDataAdapter da = new NpgsqlDataAdapter(sorgu, baglanti);
                DataSet ds = new DataSet();
                da.Fill(ds);
                dataGridView1.DataSource = ds.Tables[0];
            }
            catch (Exception ex)
            {
                MessageBox.Show("Veri getirme hatası: " + ex.Message);
            }
            finally
            {
                baglanti.Close();
            }
        }

        private void BtnAra_Click(object sender, EventArgs e)
        {
            try
            {
                int arananID = Convert.ToInt32(TxtID.Text);
                baglanti.Open();
                string sorgu = "SELECT * FROM \"SaglikVeteriner\" WHERE
\"VeterinerID\" = @arananID";
                NpgsqlDataAdapter da = new NpgsqlDataAdapter(sorgu, baglanti);
                da.SelectCommand.Parameters.AddWithValue("@arananID",
arananID);
                DataSet ds = new DataSet();
                da.Fill(ds);
                dataGridView1.DataSource = ds.Tables[0];
            }
            catch (Exception ex)
            {
                MessageBox.Show("Arama hatası: " + ex.Message);
            }
            finally
            {
                baglanti.Close();
            }
        }

        private void BtnEkle_Click(object sender, EventArgs e)
        {
            try
            {
                int veterinerID = Convert.ToInt32(TxtID.Text);
                string veterinerAd = Txtİsim.Text;
                string veterinerSoyad = TxtSoyisim.Text;
                string tedaviTipi = TxtTedaviTipi.Text;
                int tedaviUcreti = Convert.ToInt32(TxtTedaviUcreti.Text);

                baglanti.Open();
                string sorgu = "INSERT INTO
\"SaglikVeteriner\"(\"VeterinerID\", \"VeterinerAd\", \"VeterinerSoyad\",
\"TedaviTipi\", \"TedaviUcreti\") " +
                               "VALUES (@veterinerID, @veterinerAd,
@veterinerSoyad, @tedaviTipi, @tedaviUcreti)";
                NpgsqlCommand komut = new NpgsqlCommand(sorgu, baglanti);
                komut.Parameters.AddWithValue("@veterinerID", veterinerID);
                komut.Parameters.AddWithValue("@veterinerAd", veterinerAd);
                komut.Parameters.AddWithValue("@veterinerSoyad",
veterinerSoyad);
                komut.Parameters.AddWithValue("@tedaviTipi", tedaviTipi);
                komut.Parameters.AddWithValue("@tedaviUcreti", tedaviUcreti);
```

```
                komut.ExecuteNonQuery();

                MessageBox.Show("Veri başarıyla eklendi.");
            }
            catch (Exception ex)
            {
                MessageBox.Show("Veri ekleme hatası: " + ex.Message);
            }
            finally
            {
                baglanti.Close();
            }
        }

        private void BtnGuncelle_Click(object sender, EventArgs e)
        {
            try
            {
                int veterinerID = Convert.ToInt32(TxtID.Text);

                if (Txtİsim.Text != "" || TxtSoyisim.Text != "" ||
TxtTedaviTipi.Text != "" || TxtTedaviUcreti.Text != "")
                {
                    string veterinerAd = Txtİsim.Text != "" ? TxtİsimText :
null;
                    string veterinerSoyad = TxtSoyisim.Text != "" ?
TxtSoyisim.Text : null;
                    string tedaviTipi = TxtTedaviTipi.Text != "" ?
TxtTedaviTipi.Text : null;
                    int tedaviUcreti = TxtTedaviUcreti.Text != "" ?
Convert.ToInt32(TxtTedaviUcreti.Text) : -1;

                    baglanti.Open();
                    string sorgu = "UPDATE \"SaglikVeteriner\" SET ";

                    if (veterinerAd != null)
                        sorgu += "\"VeterinerAd\" = @veterinerAd, ";

                    if (veterinerSoyad != null)
                        sorgu += "\"VeterinerSoyad\" = @veterinerSoyad, ";

                    if (tedaviTipi != null)
                        sorgu += "\"TedaviTipi\" = @tedaviTipi, ";

                    if (tedaviUcreti != -1)
                        sorgu += "\"TedaviUcreti\" = @tedaviUcreti, ";

                    sorgu = sorgu.Remove(sorgu.Length - 2); // Son virgülü ve
boşluğu kaldır

                    sorgu += " WHERE \"VeterinerID\" = @veterinerID";

                    NpgsqlCommand komut = new NpgsqlCommand(sorgu, baglanti);
                    komut.Parameters.AddWithValue("@veterinerID", veterinerID);

                    if (veterinerAd != null)
                        komut.Parameters.AddWithValue("@veterinerAd",
veterinerAd);

                    if (veterinerSoyad != null)
                        komut.Parameters.AddWithValue("@veterinerSoyad",
veterinerSoyad);

                    if (tedaviTipi != null)
```

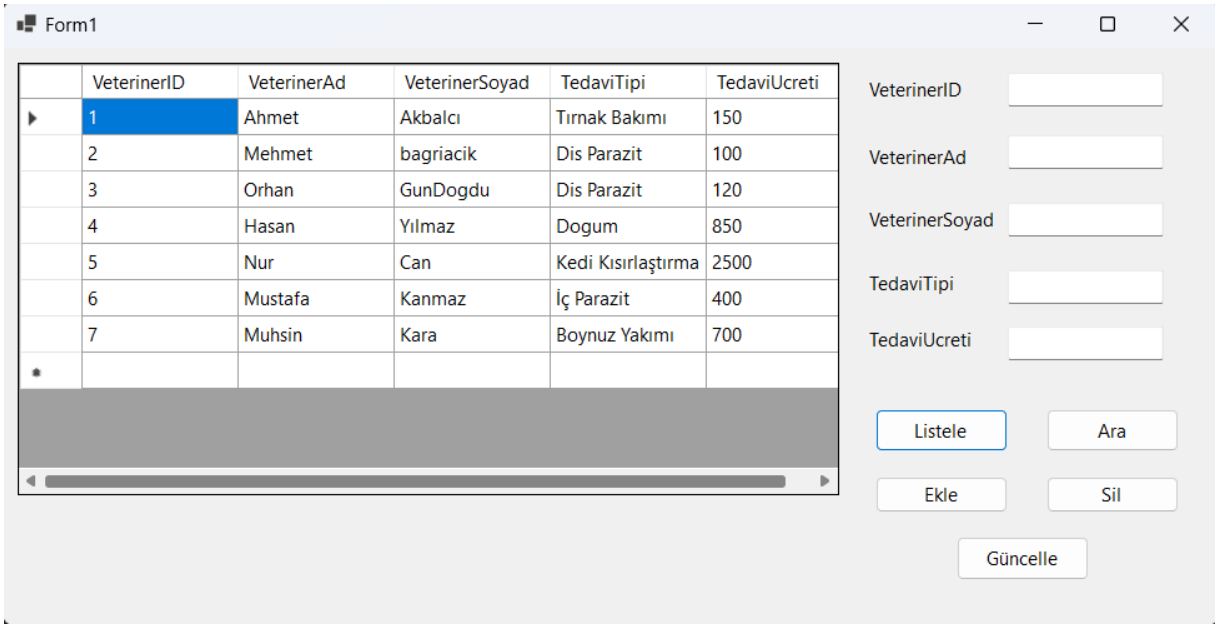
```
                        komut.Parameters.AddWithValue("@tedaviTipi",
tedaviTipi);

                    if (tedaviUcreti != -1)
                        komut.Parameters.AddWithValue("@tedaviUcreti",
tedaviUcreti);

                    komut.ExecuteNonQuery();

                    MessageBox.Show("Veri başarıyla güncellendi.");
                }
                else
                {
                    MessageBox.Show("En az bir alanı güncellemek için veri
girişi yapmalısınız.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Veri güncelleme hatası: " + ex.Message);
            }
            finally
            {
                baglanti.Close();
            }
        }

        private void BtnSil_Click(object sender, EventArgs e)
        {
            try
            {
                int veterinerID = Convert.ToInt32(TxtID.Text);

                baglanti.Open();
                string sorgu = "DELETE FROM \"SaglikVeteriner\" WHERE
\"VeterinerID\" = @veterinerID";

                NpgsqlCommand komut = new NpgsqlCommand(sorgu, baglanti);
                komut.Parameters.AddWithValue("@veterinerID", veterinerID);

                int etkilenenSatirSayisi = komut.ExecuteNonQuery();

                if (etkilenenSatirSayisi > 0)
                {
                    MessageBox.Show("Kayıt başarıyla silindi.");
                }
                else
                {
                    MessageBox.Show("Silinecek kayıt bulunamadı.");
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show("Kayıt silme hatası: " + ex.Message);
            }
            finally
            {
                baglanti.Close();
            }
        }
    }
}
```

Program.cs :

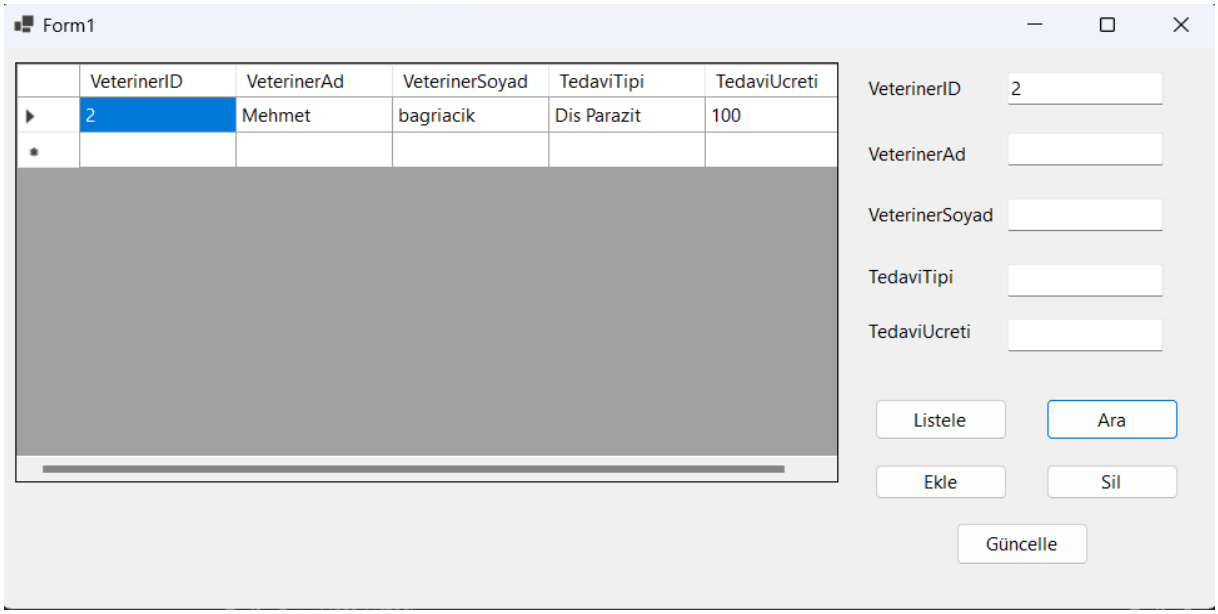
```
namespace VeriTabanı
{
    internal static class Program
    {
        /// <summary>
        ///  The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            // To customize application configuration such as set high DPI
settings or default font,
            // see https://aka.ms/applicationconfiguration.
            ApplicationConfiguration.Initialize();
            Application.Run(new Form1());
        }
    }
}
```

EKRAN GÖRÜNTÜLERİ

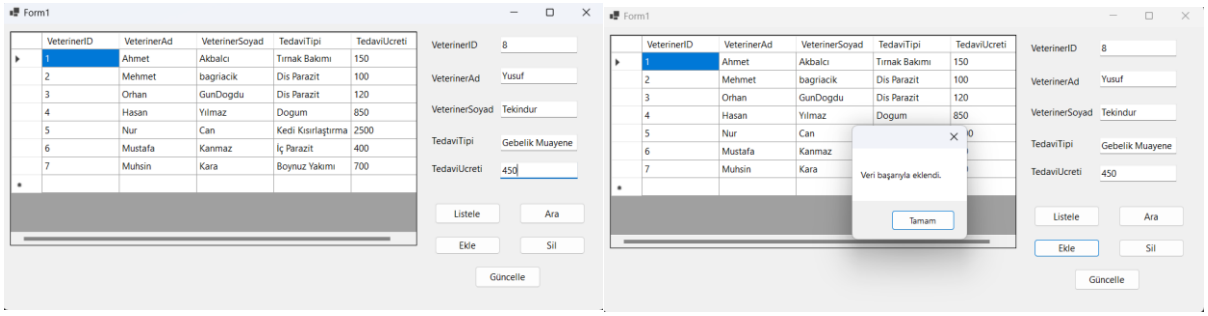
LİSTELEME

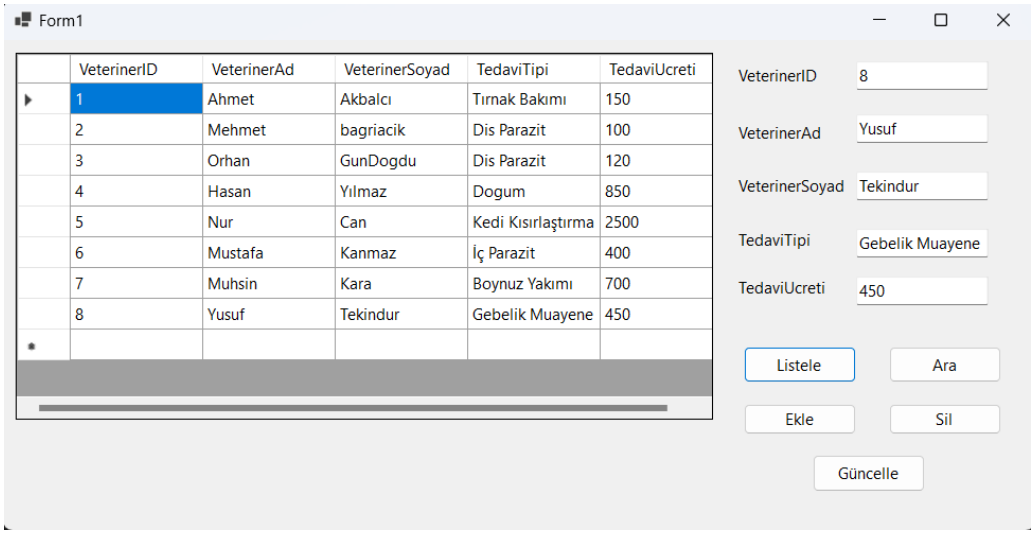


ARAMA

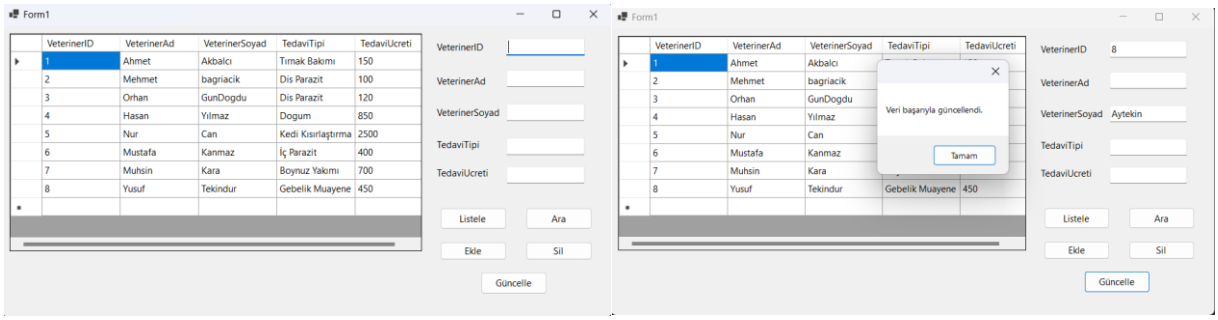


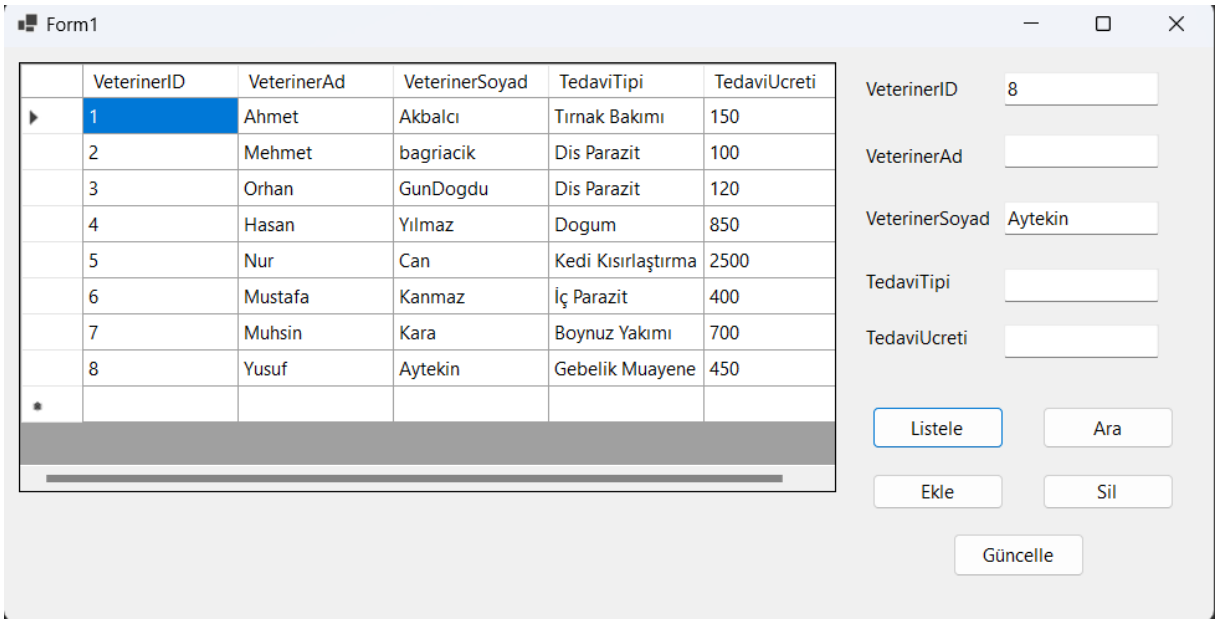
EKLEME





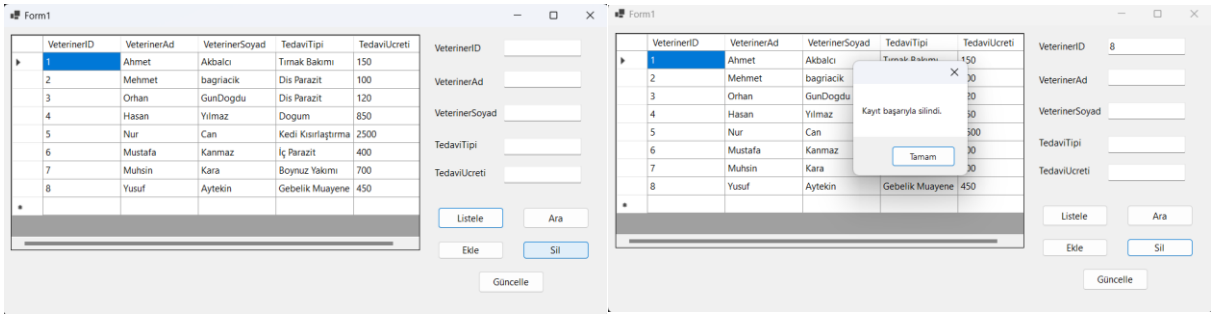
GÜNCELLEME

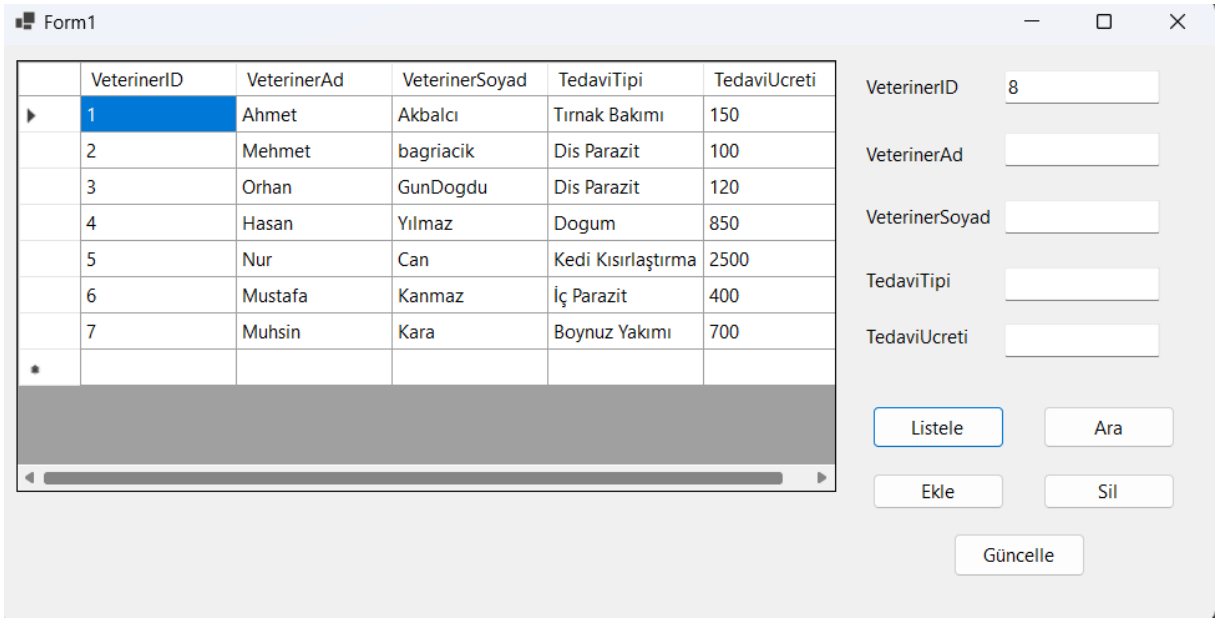




VeterinerID	VeterinerAd	VeterinerSoyad	TedaviTipi	TedaviUcreti
1	Ahmet	Akbalcı	Tırnak Bakımı	150
2	Mehmet	bagriacik	Dis Parazit	100
3	Orhan	GunDogdu	Dis Parazit	120
4	Hasan	Yılmaz	Dogum	850
5	Nur	Can	Kedi Kısırlaştırma	2500
6	Mustafa	Kanmaz	İç Parazit	400
7	Muhsin	Kara	Boynuz Yakımı	700
8	Yusuf	Aytekin	Gebelik Muayene	450

SİLME





VeterinerID	VeterinerAd	VeterinerSoyad	TedaviTipi	TedaviUcreti
1	Ahmet	Akbalcı	Tırnak Bakımı	150
2	Mehmet	bagriacik	Dis Parazit	100
3	Orhan	GunDogdu	Dis Parazit	120
4	Hasan	Yılmaz	Dogum	850
5	Nur	Can	Kedi Kısırlaştırma	2500
6	Mustafa	Kanmaz	İç Parazit	400
7	Muhsin	Kara	Boynuz Yakımı	700