SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ VERİTABANI YÖNETİM SİSTEMLERİ

DERSİ PROJE ÖDEVİ (2023-2024 Güz Dönemi)

Ömer Zahid Kara

g191210070 - 2C

omer.kara7@ogr.sakarya.edu.tr

İÇERİK

- Uygulamanın kısa tanıtımı, iş kuralları , ilişkisel şema
- Varlık Bağıntı Modeli
- SQL İfadeleri
- Saklı yordam ve tetikleyiciler
- Uygulamanın kaynak kodları

Uygulama Tanıtımı

3 çeşit iş görevine sahip (Balıkçılık, ticari yük taşımacılığı ve yolcu taşıma) gemi seferlerinin bilgilerini tutan veri tabanı için sefer düzenleyici programdır. Sefere çıkan geminin özellikleri,seyir rotası, ne yapacağı ve kullandığı limanların özellikleri gibi verilerin yönetildiği bir programdır. Veritabanı yönetim sistemi olarak PostgreSQL kullanılmış, uygulama ise C# Windows Form ile oluşturulmuştur.

İŞ KURALLARI

- Seferlerin sefer kodu(kodu 50 ile başlar ve yanına rakam alır, ör: 5023 veya 501), sahip oldukları geminin kodu(kodu 60 ile başlar ve yanına rakam alır, ör: 6023 veya 601), seferin iş kodu(kodu 30 ile başlar ve yanına rakam alır, ör: 3023 veya 301) ve rota(70---) kodu vardır.

- Gemilerin gemi kodu, adı, türü(handysize,handymax,supramax,panamax,post-panamax,capesize,VLOC), ağırlığı(DWT), azami sürat bilgisi(km/saat) vardır.
- Personllerin kendi id(kodu 10 ile başlar ve yanına rakam alır, ör: 1023 veya 101), isim, soyad, yaş, bulunduklarının geminin kodu ve cinsiyet bilgileri vardır.
- Personeller yönetici(Gemi kaptanı, kaptan yardımcısı, şirket üst düzey temsilcisi) ve iş personeli(Mühendis, gemi teknisyeni, aşçı, temizlikçi, danışman, güvenlik, iş görevlisi) olarak ikiye ayrılır.
- İşler iş kodu (30---), adı(yük,yolcu,balıkçı), süresi(gün), ve işin sahibi firmanın ad bilgilerine sahiptir.
- İşler deniz mahsülü üretimi, yolcu taşıma ve yük taşıma olarak üçe ayrılır.
- Deniz mahsülü üretiminde üretimin kilogram cinsinden hedef bilgisi ve iş kodu vardır.
- Yolcu taşıma işinin yolcu sayısı bilgisi ve iş kodu vardır.
- Yük taşıma işi, yük miktarı konteyner sayısı cinsinden ve taşıdığı yükün adı bilgilerine sahiptir.
- İş sahibi şirketin adı,id si, CEO bilgisi ve ülke bilgisi vardır.
- Ülkelerin adı, nüfusu ve bulunduğu kıta vardır. Birden fazla kıtada toprağı olan ülkenin ağırlıklı olarak bulunduğu kıta önceliklidir.
- Rotaların kodu(70--), mesafe bilgisi(km), gemisinin ortalama seyir sürati(km/s), durak limanları idleri(11--), kalkış yapacağı liman id(11--), ve varış yapacağı liman idsi(11--), vardır.
- Rotaların varış limanları varış olarak saklanırken kalkacakları liman ve tarihi tutan kayıt kalkış kayıt olarak saklanır. Varış kaydının ve kalkış kaydının id si vardır. Kalkış kaydının kalkış tarihi, varış kaydının varış tarihi vardır. Bu kayıtlarda liman bilgileri de vardır.
- Limanların kodu(11--), adı, ülkesi, gemi kapasitesi vardır.
- Bir seferin bir rotası, bir gemisi, bir iş tanımı olur.
- Bir geminin, bir seferi, en az bir personeli, en çok çok personeli olur.
- Bir personel ya yönetici personeli ya da iş personeli olabilir.
- Bir işin bir yönetici şirketi vardır.
- Bir firmanın hiç işi olmayabilir ya da çok işi olabilir.
- Bir iş ya deniz mahsülü üretimi ya yolcu taşımacılığı ya da yük taşımacılığı olabilir.
- Bir şirketin sadece bir ülkesi olur.
- Bir ülkenin hiç şirketi ve limanı olmayabilirken en çok çok şirketi ve limanı olabilir.
- Bir limanın bir ülkesi olur.
- Bir rotanın en az bir en çok bir kalkış ve varış kaydı olur.
- Bir varış kaydı ve kalkış kaydının bir rotası ve limanı olur.

- Bir limanın en az hiç en çok çok kalkış kaydı ve varış kaydı olabilir.

İLİŞKİSEL ŞEMA

-sefer(**sefer_kodu:bigint**, gemi_kodu:bigint, is_kodu: bigint, rota_kodu:bigint)

-rota(**rota_kodu:bigint**, mesafe:integer, ort_seyir_surati:integer)

-gemi(**gemi_kodu:bigint**, gemi_adi:varchar**,** turu:varchar, agirligi:integer, azami_surati:integer)

-is(**is_kodu:bigint** , sirket_adi:varchar, is_adi:varchar, is_suresi: integer)

-personel(**personel_id:bigint**,gemi_kodu:bigint, isim:varchar, soyad:varchar, yas:integer, cinsiyeti:varchar)

- yonetici_personel**(personel_id:bigint**, uzmanlik_adi:varchar)

-is_personeli(**personel_id:bigint**, **,** uzmanlik_adi:varchar)

-sirket(**sirket_id:integer**,sirked_adi= varchar, sirket_ceo:varchar ,ulke_adi:varchar)

-ülke**(ulke_adi:varhchar**, nufus:bigint, kitasi:varchar)

-liman(**liman_kodu:bigint ,**liman_adi:varchar**,** ulke_adi:varchar, gemi_kapasitesi:integer)

-deniz_mahsulu_uretim(**is_kodu:bigint,** uretim_hedefi:integer)

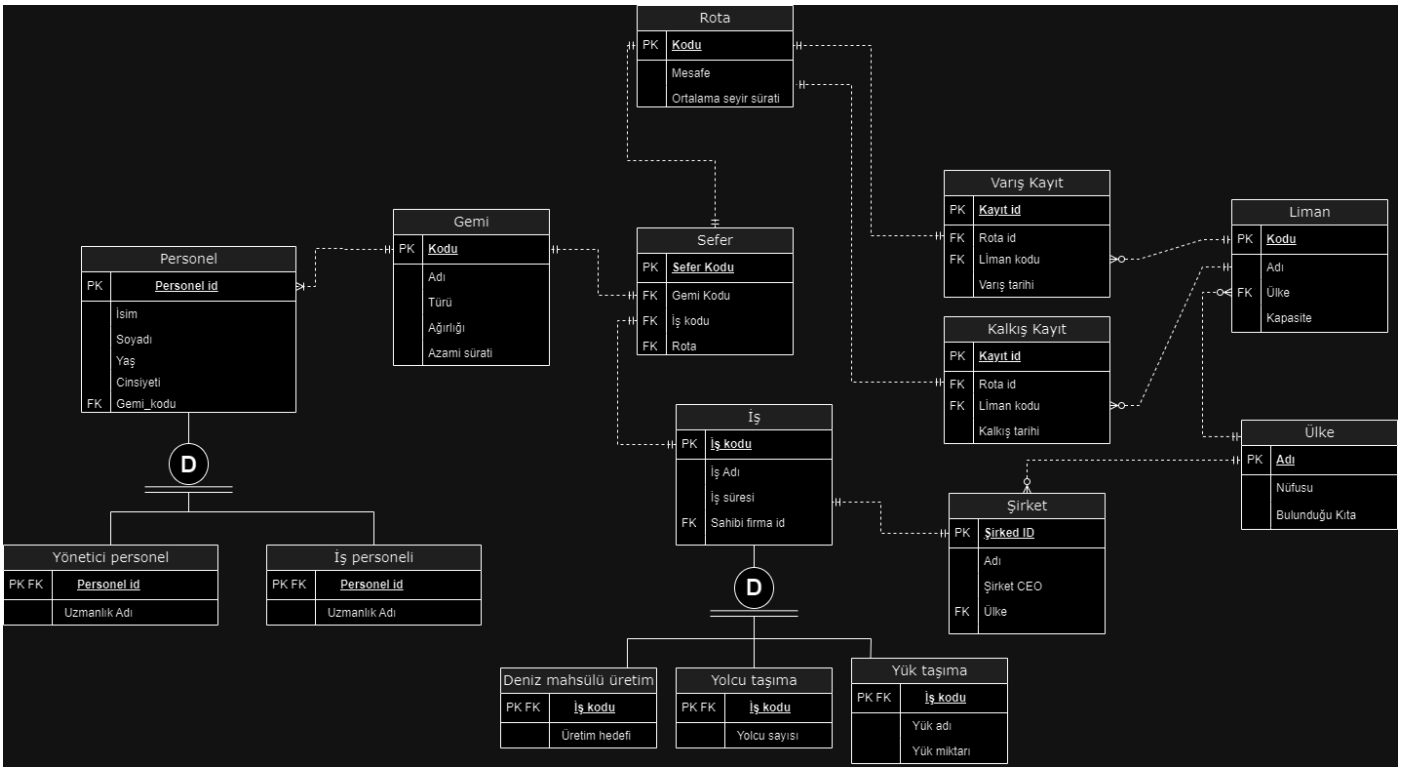
-yolcu_tasima**(is_kodu:bigint,** yolcu_sayisi:integer**)**

-yuk_tasima(**is_kodu:bigint,** yuk_adi:varchar, yuk_sayisi:integer)

-varis_kayit(**varis_kayit_id:integer**, rota_kodu:bigint, liman_kodu:bigint, varis_tarihi: date)

-kalkis_kayit(**kalkis_kayit_id:integer**, rota_kodu:bigint, liman_kodu:bigint, kalkis_tarihi: date)

VARLIK BAĞINTI DİYAGRAMI



SQL İFADELERİ

-- Table: public.deniz_mahsulu_uretim

-- DROP TABLE public.deniz_mahsulu_uretim;

```
CREATE TABLE IF NOT EXISTS public.deniz_mahsulu_uretim
(
    is_kodu bigint NOT NULL,
    uretim_hedefi integer,
    CONSTRAINT "deniz_mahsulu_uretimPK" PRIMARY KEY (is_kodu),
    CONSTRAINT "yolcu_Tasima" FOREIGN KEY (is_kodu)
        REFERENCES public.is_ (is_kodu) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

```
TABLESPACE pg_default;

ALTER TABLE public.deniz_mahsulu_uretim
    OWNER to postgres;
-- Constraint: deniz_mahsulu_uretimPK

-- ALTER TABLE public.deniz_mahsulu_uretim DROP CONSTRAINT "deniz_mahsulu_uretimPK";

ALTER TABLE public.deniz_mahsulu_uretim
    ADD CONSTRAINT "deniz_mahsulu_uretimPK" PRIMARY KEY (is_kodu);
-- Constraint: yolcu_Tasima

-- ALTER TABLE public.deniz_mahsulu_uretim DROP CONSTRAINT "yolcu_Tasima";

ALTER TABLE public.deniz_mahsulu_uretim
    ADD CONSTRAINT "yolcu_Tasima" FOREIGN KEY (is_kodu)
    REFERENCES public.is_ (is_kodu) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
-- Table: public.gemi

-- DROP TABLE public.gemi;

CREATE TABLE IF NOT EXISTS public.gemi
(
    gemi_kodu bigint NOT NULL,
    adi character varying(20) COLLATE pg_catalog."default",
    turu character varying(20) COLLATE pg_catalog."default",
    agirligi integer,
```

```
    azami_surati integer,
    CONSTRAINT gemi_pkey PRIMARY KEY (gemi_kodu),
    CONSTRAINT gemi_unq UNIQUE (gemi_kodu)
)

TABLESPACE pg_default;

ALTER TABLE public.gemi
    OWNER to postgres;
-- Constraint: gemi_pkey

-- ALTER TABLE public.gemi DROP CONSTRAINT gemi_pkey;

ALTER TABLE public.gemi
    ADD CONSTRAINT gemi_pkey PRIMARY KEY (gemi_kodu);
-- Constraint: gemi_unq

-- ALTER TABLE public.gemi DROP CONSTRAINT gemi_unq;

ALTER TABLE public.gemi
    ADD CONSTRAINT gemi_unq UNIQUE (gemi_kodu);
-- Table: public.is_

-- DROP TABLE public.is_;

CREATE TABLE IF NOT EXISTS public.is_
(
    is_kodu bigint NOT NULL,
    is_adi character varying(20) COLLATE pg_catalog."default",
    is_suresi integer,
```

```
    sahibi_firma_id integer NOT NULL DEFAULT nextval('is__sahibi_firma_id_seq'::regclass),
    CONSTRAINT is_pkey PRIMARY KEY (is_kodu),
    CONSTRAINT is_unq UNIQUE (is_kodu),
    CONSTRAINT isfg FOREIGN KEY (sahibi_firma_id)
        REFERENCES public.sirket (sirket_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.is_
    OWNER to postgres;
-- Index: fki_is_fkkkkkk

-- DROP INDEX public.fki_is_fkkkkkk;

CREATE INDEX fki_is_fkkkkkk
    ON public.is_ USING btree
    (sahibi_firma_id ASC NULLS LAST)
    TABLESPACE pg_default;
-- Constraint: is_pkey

-- ALTER TABLE public.is_ DROP CONSTRAINT is_pkey;

ALTER TABLE public.is_
    ADD CONSTRAINT is_pkey PRIMARY KEY (is_kodu);
-- Constraint: is_unq
```

```
-- ALTER TABLE public.is_ DROP CONSTRAINT is_unq;

ALTER TABLE public.is_
    ADD CONSTRAINT is_unq UNIQUE (is_kodu);
-- Constraint: isfg

-- ALTER TABLE public.is_ DROP CONSTRAINT isfg;

ALTER TABLE public.is_
    ADD CONSTRAINT isfg FOREIGN KEY (sahibi_firma_id)
    REFERENCES public.sirket (sirket_id) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
-- Table: public.is_personeli

-- DROP TABLE public.is_personeli;

CREATE TABLE IF NOT EXISTS public.is_personeli
(
    personel_id bigint NOT NULL,
    uzmanlik character varying(25) COLLATE pg_catalog."default",
    CONSTRAINT "is_personelPK" PRIMARY KEY (personel_id),
    CONSTRAINT "isPersonel" FOREIGN KEY (personel_id)
        REFERENCES public.personel (personel_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;
```

```
ALTER TABLE public.is_personeli
    OWNER to postgres;
-- Constraint: isPersonel

-- ALTER TABLE public.is_personeli DROP CONSTRAINT "isPersonel";

ALTER TABLE public.is_personeli
    ADD CONSTRAINT "isPersonel" FOREIGN KEY (personel_id)
    REFERENCES public.personel (personel_id) MATCH SIMPLE
    ON UPDATE CASCADE
    ON DELETE CASCADE;
-- Constraint: is_personelPK

-- ALTER TABLE public.is_personeli DROP CONSTRAINT "is_personelPK";

ALTER TABLE public.is_personeli
    ADD CONSTRAINT "is_personelPK" PRIMARY KEY (personel_id);

-- Table: public.kalkis_kayit

-- DROP TABLE public.kalkis_kayit;

CREATE TABLE IF NOT EXISTS public.kalkis_kayit
(
    rota_kodu bigint NOT NULL,
    liman_kodu bigint NOT NULL,
    kalkis_kayit_id integer NOT NULL DEFAULT nextval('kalkis_kayit_kalkis_kayit_id_seq'::regclass),
    kalkis_tarihi character varying(25) COLLATE pg_catalog."default",
    CONSTRAINT kalkis_kayit_pkey PRIMARY KEY (kalkis_kayit_id),
```

```
    CONSTRAINT rota_unq UNIQUE (rota_kodu),
    CONSTRAINT kalkis_rota_fk FOREIGN KEY (rota_kodu)
        REFERENCES public.rota (kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT liman_kalkis_fk FOREIGN KEY (liman_kodu)
        REFERENCES public.liman (kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.kalkis_kayit
    OWNER to postgres;
-- Index: fki_kalkis_liman_fk

-- DROP INDEX public.fki_kalkis_liman_fk;

CREATE INDEX fki_kalkis_liman_fk
    ON public.kalkis_kayit USING btree
    (liman_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_kalkis_rota_fk

-- DROP INDEX public.fki_kalkis_rota_fk;

CREATE INDEX fki_kalkis_rota_fk
```

```
    ON public.kalkis_kayit USING btree
    (rota_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Constraint: kalkis_kayit_pkey

-- ALTER TABLE public.kalkis_kayit DROP CONSTRAINT kalkis_kayit_pkey;

ALTER TABLE public.kalkis_kayit
    ADD CONSTRAINT kalkis_kayit_pkey PRIMARY KEY (kalkis_kayit_id);
-- Constraint: kalkis_rota_fk

-- ALTER TABLE public.kalkis_kayit DROP CONSTRAINT kalkis_rota_fk;

ALTER TABLE public.kalkis_kayit
    ADD CONSTRAINT kalkis_rota_fk FOREIGN KEY (rota_kodu)
    REFERENCES public.rota (kodu) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
-- Constraint: liman_kalkis_fk

-- ALTER TABLE public.kalkis_kayit DROP CONSTRAINT liman_kalkis_fk;

ALTER TABLE public.kalkis_kayit
    ADD CONSTRAINT liman_kalkis_fk FOREIGN KEY (liman_kodu)
    REFERENCES public.liman (kodu) MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
-- Constraint: rota_unq
```

```
-- ALTER TABLE public.kalkis_kayit DROP CONSTRAINT rota_unq;

ALTER TABLE public.kalkis_kayit
    ADD CONSTRAINT rota_unq UNIQUE (rota_kodu);
-- Table: public.liman

-- DROP TABLE public.liman;

CREATE TABLE IF NOT EXISTS public.liman
(
    kodu bigint NOT NULL,
    adi character varying(20) COLLATE pg_catalog."default",
    ulke_adi character varying(20) COLLATE pg_catalog."default",
    gemi_kapasitesi integer,
    CONSTRAINT liman_pkey PRIMARY KEY (kodu),
    CONSTRAINT liman_ulke_fk FOREIGN KEY (ulke_adi)
        REFERENCES public.ulke (ulke_adi) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.liman
    OWNER to postgres;
-- Index: fki_liman_ulke_fk

-- DROP INDEX public.fki_liman_ulke_fk;
```

```
CREATE INDEX fki_liman_ulke_fk
    ON public.liman USING btree
    (ulke_adi COLLATE pg_catalog."default" ASC NULLS LAST)
    TABLESPACE pg_default;
-- Table: public.personel

-- DROP TABLE public.personel;

CREATE TABLE IF NOT EXISTS public.personel
(
    personel_id bigint NOT NULL,
    isim character varying(20) COLLATE pg_catalog."default",
    soyad character varying(20) COLLATE pg_catalog."default",
    yas integer,
    cinsiyet character varying(20) COLLATE pg_catalog."default",
    gemi_kodu bigint,
    CONSTRAINT personel_pkey PRIMARY KEY (personel_id),
    CONSTRAINT personel_gemi_fk FOREIGN KEY (gemi_kodu)
        REFERENCES public.gemi (gemi_kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.personel
    OWNER to postgres;
-- Index: fki_personel_gemi_fk
```

```
-- DROP INDEX public.fki_personel_gemi_fk;

CREATE INDEX fki_personel_gemi_fk
    ON public.personel USING btree
    (gemi_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Table: public.rota

-- DROP TABLE public.rota;

CREATE TABLE IF NOT EXISTS public.rota
(
    kodu bigint NOT NULL,
    mesafe integer,
    ort_seyir_surati integer,
    CONSTRAINT rota_pkey PRIMARY KEY (kodu),
    CONSTRAINT rota_unq_self UNIQUE (kodu)
)

TABLESPACE pg_default;

ALTER TABLE public.rota
    OWNER to postgres;
-- Table: public.sefer

-- DROP TABLE public.sefer;

CREATE TABLE IF NOT EXISTS public.sefer
(
```

```
    sefer_kodu bigint NOT NULL,
    gemi_kodu bigint NOT NULL,
    is_kodu bigint,
    rota_kodu bigint NOT NULL,
    CONSTRAINT sefer_pkey PRIMARY KEY (sefer_kodu),
    CONSTRAINT sefer_uqn UNIQUE (sefer_kodu, is_kodu, rota_kodu),
    CONSTRAINT uniq_gemi UNIQUE (gemi_kodu),
    CONSTRAINT uniq_is UNIQUE (is_kodu),
    CONSTRAINT uniq_rota UNIQUE (rota_kodu),
    CONSTRAINT is_sefer_unq FOREIGN KEY (is_kodu)
        REFERENCES public.is_ (is_kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT sefer_gemi_fk FOREIGN KEY (gemi_kodu)
        REFERENCES public.gemi (gemi_kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT sefer_rota_fk FOREIGN KEY (rota_kodu)
        REFERENCES public.rota (kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.sefer
    OWNER to postgres;
```

```
-- Index: fki_sefer_gemi_fk

-- DROP INDEX public.fki_sefer_gemi_fk;

CREATE INDEX fki_sefer_gemi_fk
    ON public.sefer USING btree
    (gemi_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_sefer_is_fk

-- DROP INDEX public.fki_sefer_is_fk;

CREATE INDEX fki_sefer_is_fk
    ON public.sefer USING btree
    (is_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_sefer_rota_fk

-- DROP INDEX public.fki_sefer_rota_fk;

CREATE INDEX fki_sefer_rota_fk
    ON public.sefer USING btree
    (rota_kodu ASC NULLS LAST)
    TABLESPACE pg_default;

-- Trigger: trg_gemi_kullanim_durumu

-- DROP TRIGGER trg_gemi_kullanim_durumu ON public.sefer;

CREATE TRIGGER trg_gemi_kullanim_durumu
```

```
    AFTER INSERT OR UPDATE
    ON public.sefer
    FOR EACH ROW
    EXECUTE FUNCTION public.gemi_kullanim_durumu();

-- Trigger: trg_gemi_kullanim_durumu_silme

-- DROP TRIGGER trg_gemi_kullanim_durumu_silme ON public.sefer;

CREATE TRIGGER trg_gemi_kullanim_durumu_silme
    AFTER DELETE
    ON public.sefer
    FOR EACH ROW
    EXECUTE FUNCTION public.gemi_kullanim_durumu_silme();

-- Trigger: trg_sefer_kodu_guncelle

-- DROP TRIGGER trg_sefer_kodu_guncelle ON public.sefer;

CREATE TRIGGER trg_sefer_kodu_guncelle
    BEFORE INSERT
    ON public.sefer
    FOR EACH ROW
    EXECUTE FUNCTION public.sefer_kodu_guncelle();

-- Trigger: trg_sil_sefer

-- DROP TRIGGER trg_sil_sefer ON public.sefer;

CREATE TRIGGER trg_sil_sefer
```

```
    AFTER DELETE
    ON public.sefer
    FOR EACH ROW
    EXECUTE FUNCTION public.sil_sefer();
-- Table: public.sirket

-- DROP TABLE public.sirket;

CREATE TABLE IF NOT EXISTS public.sirket
(
    sirket_ceo character varying(25) COLLATE pg_catalog."default",
    ulke_adi character varying(25) COLLATE pg_catalog."default" NOT NULL,
    sirket_adi character varying(25) COLLATE pg_catalog."default",
    sirket_id integer NOT NULL DEFAULT nextval('sirket_sirket_id_seq'::regclass),
    CONSTRAINT sirket_pkey PRIMARY KEY (sirket_id),
    CONSTRAINT unique_sirket_adi UNIQUE (sirket_adi),
    CONSTRAINT sirket_ulke_fk FOREIGN KEY (ulke_adi)
        REFERENCES public.ulke (ulke_adi) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.sirket
    OWNER to postgres;
-- Index: fki_sirket_ulke_fk

-- DROP INDEX public.fki_sirket_ulke_fk;
```

```
CREATE INDEX fki_sirket_ulke_fk
    ON public.sirket USING btree
    (ulke_adi COLLATE pg_catalog."default" ASC NULLS LAST)
    TABLESPACE pg_default;
-- Table: public.ulke

-- DROP TABLE public.ulke;

CREATE TABLE IF NOT EXISTS public.ulke
(
    ulke_adi character varying COLLATE pg_catalog."default" NOT NULL,
    nufus bigint,
    kitasi character varying COLLATE pg_catalog."default",
    CONSTRAINT "ülke_pkey" PRIMARY KEY (ulke_adi),
    CONSTRAINT uniq_ule UNIQUE (ulke_adi)
)

TABLESPACE pg_default;

ALTER TABLE public.ulke
    OWNER to postgres;
-- Table: public.varis_kayit

-- DROP TABLE public.varis_kayit;

CREATE TABLE IF NOT EXISTS public.varis_kayit
(
    rota_kodu bigint NOT NULL,
    liman_kodu bigint NOT NULL,
```

```
    varis_kayit_id integer NOT NULL DEFAULT nextval('varis_kayit_varis_kayit_id_seq'::regclass),
    varis_tarihi character varying(25) COLLATE pg_catalog."default",
    CONSTRAINT varis_kayit_pkey PRIMARY KEY (varis_kayit_id),
    CONSTRAINT uniq_rotaa UNIQUE (rota_kodu),
    CONSTRAINT varis_liman_fk FOREIGN KEY (liman_kodu)
        REFERENCES public.liman (kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT varis_rota_fk FOREIGN KEY (rota_kodu)
        REFERENCES public.rota (kodu) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public.varis_kayit
    OWNER to postgres;
-- Index: fki_varis_liman_fk

-- DROP INDEX public.fki_varis_liman_fk;

CREATE INDEX fki_varis_liman_fk
    ON public.varis_kayit USING btree
    (liman_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_varis_rota_fk
```

```
-- DROP INDEX public.fki_varis_rota_fk;

CREATE INDEX fki_varis_rota_fk
    ON public.varis_kayit USING btree
    (rota_kodu ASC NULLS LAST)
    TABLESPACE pg_default;
-- Table: public.yolcu_tasima

-- DROP TABLE public.yolcu_tasima;

CREATE TABLE IF NOT EXISTS public.yolcu_tasima
(
    is_kodu bigint NOT NULL,
    "yolcu_Sayisi" integer,
    CONSTRAINT "yolcu_tasimaPK" PRIMARY KEY (is_kodu),
    CONSTRAINT "yolcu_Tasima" FOREIGN KEY (is_kodu)
        REFERENCES public.is_ (is_kodu) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;

ALTER TABLE public.yolcu_tasima
    OWNER to postgres;
-- Table: public.yonetici_personel

-- DROP TABLE public.yonetici_personel;

CREATE TABLE IF NOT EXISTS public.yonetici_personel
```

```
(
    personel_id bigint NOT NULL,
    uzmanlik character varying(25) COLLATE pg_catalog."default",
    CONSTRAINT "yonetici_personelPK" PRIMARY KEY (personel_id),
    CONSTRAINT "yoneticiPersonel" FOREIGN KEY (personel_id)
        REFERENCES public.personel (personel_id) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)

TABLESPACE pg_default;

ALTER TABLE public.yonetici_personel
    OWNER to postgres;
-- Table: public.yuk_tasima

-- DROP TABLE public.yuk_tasima;

CREATE TABLE IF NOT EXISTS public.yuk_tasima
(
    is_kodu bigint NOT NULL,
    yuk_adi character varying(25) COLLATE pg_catalog."default",
    yuk_miktari integer,
    CONSTRAINT "yuk_tasimaPK" PRIMARY KEY (is_kodu),
    CONSTRAINT "yuk_Tasima" FOREIGN KEY (is_kodu)
        REFERENCES public.is_ (is_kodu) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
)
```

```
TABLESPACE pg_default;


ALTER TABLE public.yuk_tasima
    OWNER to postgres;
```

Fonksiyon 1: Sefer kaydı silindiğinde bir trigger ile sırasına göre ilişkili tablodaki verinin silinmesi için yönetilme yapılır. Bu fonksiyon iş tablosundaki ilgili veriyi silmeye yarar.

```
-- FUNCTION: public.sil_is(bigint)


-- DROP FUNCTION public.sil_is(bigint);


CREATE OR REPLACE FUNCTION public.sil_is(
            _is_kodu bigint)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
BEGIN
  DELETE FROM public.is_ WHERE is_kodu = _is_kodu;
END;
$BODY$;


ALTER FUNCTION public.sil_is(bigint)
    OWNER TO postgres;
```

Fonksiyon 2: Sefer kaydı silindiğinde bir trigger ile sırasına göre ilişkili tablodaki verinin silinmesi için yönetilme yapılır. Bu fonksiyon kalkis_kayit tablosundaki ilgili veriyi silmeye yarar.

```
-- FUNCTION: public.sil_kalkis_kayit(bigint)


-- DROP FUNCTION public.sil_kalkis_kayit(bigint);


CREATE OR REPLACE FUNCTION public.sil_kalkis_kayit(
```

```
        _rota_kodu bigint)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
BEGIN
   DELETE FROM public.kalkis_kayit WHERE rota_kodu = _rota_kodu;
END;
$BODY$;


ALTER FUNCTION public.sil_kalkis_kayit(bigint)
    OWNER TO postgres;
```

Fonksiyon 3: Sefer kaydı silindiğinde bir trigger ile sırasına göre ilişkili tablodaki verinin silinmesi için yönetilme yapılır. Bu fonksiyon rota_ tablosundaki ilgili veriyi silmeye yarar.

```
-- FUNCTION: public.sil_rota(bigint)


-- DROP FUNCTION public.sil_rota(bigint);


CREATE OR REPLACE FUNCTION public.sil_rota(
        _rota_kodu bigint)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
BEGIN
   DELETE FROM public.rota WHERE kodu = _rota_kodu;
END;
$BODY$;
```

```
ALTER FUNCTION public.sil_rota(bigint)
    OWNER TO postgres;
```

Fonksiyon 4: Sefer kaydı silindiğinde bir trigger ile sırasına göre ilişkili tablodaki verinin silinmesi için yönetilme yapılır. Bu fonksiyon varis_kayit tablosundaki ilgili veriyi silmeye yarar.

```
-- FUNCTION: public.sil_varis_kayit(bigint)


-- DROP FUNCTION public.sil_varis_kayit(bigint);


CREATE OR REPLACE FUNCTION public.sil_varis_kayit(
        _rota_kodu bigint)
    RETURNS void
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE PARALLEL UNSAFE
AS $BODY$
BEGIN
  DELETE FROM public.varis_kayit WHERE rota_kodu = _rota_kodu;
END;
$BODY$;


ALTER FUNCTION public.sil_varis_kayit(bigint)
    OWNER TO postgres;
```

Trigger 1: Bu trigger form uygulamasında bir sefer eklenirse eklenen seferdeki geminin adını (Kullanılıyor) olarak değiştirir. Kullanıcının o gemiyi seçmemesi beklenir.

```
-- FUNCTION: public.gemi_kullanim_durumu()


-- DROP FUNCTION public.gemi_kullanim_durumu();


CREATE OR REPLACE FUNCTION public.gemi_kullanim_durumu()
    RETURNS trigger
    LANGUAGE 'plpgsql'
```

```
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    IF TG_OP = 'INSERT' OR TG_OP = 'UPDATE' THEN
        UPDATE public.gemi
        SET adi = CONCAT(adi, ' (Kullanımda)')
        WHERE gemi_kodu = NEW.gemi_kodu;
    END IF;
    RETURN NEW;
END;
$BODY$;

ALTER FUNCTION public.gemi_kullanim_durumu()
    OWNER TO postgres;
```

Trigger 2: Trigger 1 in sefer silme durumunda tersini yapar. Silinen seferin gemisinin adını düzeltir.

```
-- FUNCTION: public.gemi_kullanim_durumu_silme()

-- DROP FUNCTION public.gemi_kullanim_durumu_silme();

CREATE OR REPLACE FUNCTION public.gemi_kullanim_durumu_silme()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    IF TG_OP = 'DELETE' THEN
        UPDATE public.gemi
        SET adi = REPLACE(adi, ' (Kullanımda)', '')
```

```
      WHERE gemi_kodu = OLD.gemi_kodu;

  END IF;

  RETURN OLD;

END;

$BODY$;


ALTER FUNCTION public.gemi_kullanim_durumu_silme()

  OWNER TO postgres;
```

Trigger 3: Sefer kodunu insert işlemi sonrası 100 ile çarparak ayrıştırmayı hedefler.

```
-- FUNCTION: public.sefer_kodu_guncelle()


-- DROP FUNCTION public.sefer_kodu_guncelle();


CREATE OR REPLACE FUNCTION public.sefer_kodu_guncelle()

  RETURNS trigger

  LANGUAGE 'plpgsql'

  COST 100

  VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

  NEW.sefer_kodu = NEW.sefer_kodu * 100;

  RETURN NEW;

END;

$BODY$;


ALTER FUNCTION public.sefer_kodu_guncelle()

  OWNER TO postgres;
```

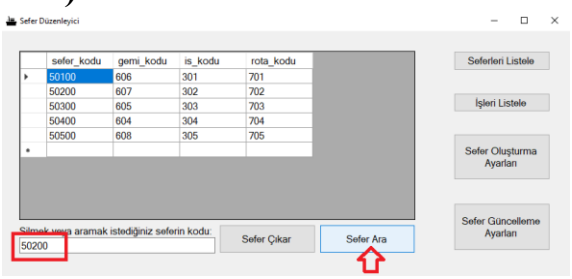
Trigger4: Sefer silme işlemi sonrası veri tabanı optimizasyonunu sağlamak amacıyla tasarlanmış 4 fonksiyonu önem sırasına göre çağırır. Önce iş kaydı silinir. Onu kalkış ve varış kaydı izler. En son rota silinir.

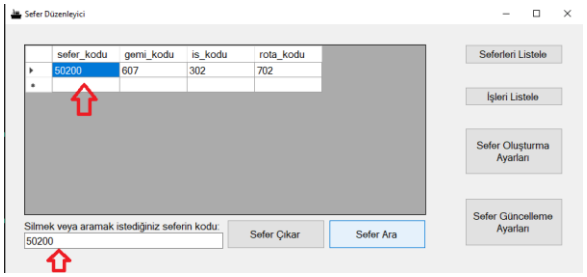
```
-- FUNCTION: public.sil_sefer()
```

```
-- DROP FUNCTION public.sil_sefer();

CREATE OR REPLACE FUNCTION public.sil_sefer()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
   -- İş'i sil
   PERFORM sil_is(OLD.is_kodu);
   -- Önce kalkış kayıtlarını sil
   PERFORM sil_kalkis_kayit(OLD.rota_kodu);

   -- Ardından varış kayıtlarını sil
   PERFORM sil_varis_kayit(OLD.rota_kodu);

   -- Rota'yı sil
   PERFORM sil_rota(OLD.rota_kodu);

   RETURN OLD;
END;
$BODY$;

ALTER FUNCTION public.sil_sefer()
    OWNER TO postgres;
```

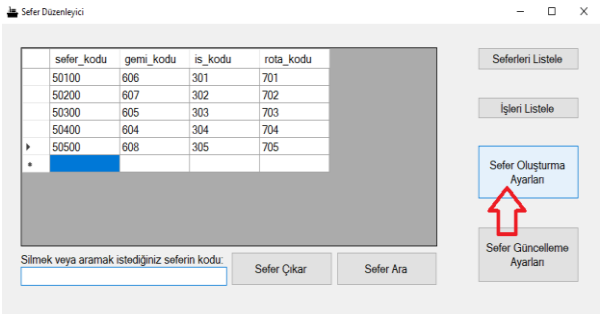
UYGULAMAYA AİT EKRAN GÖRÜNTÜLERİ

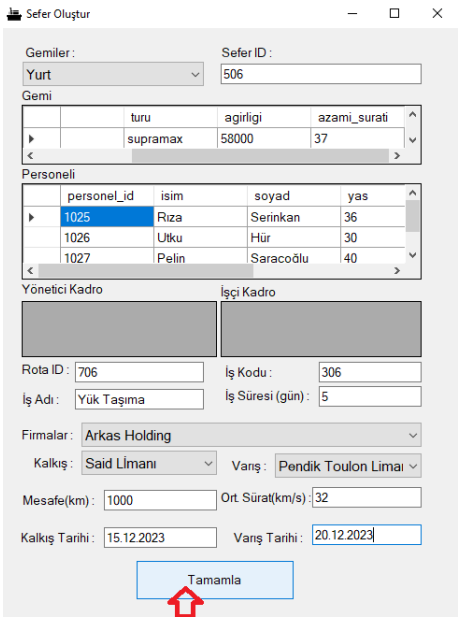
1) Sefer Arama



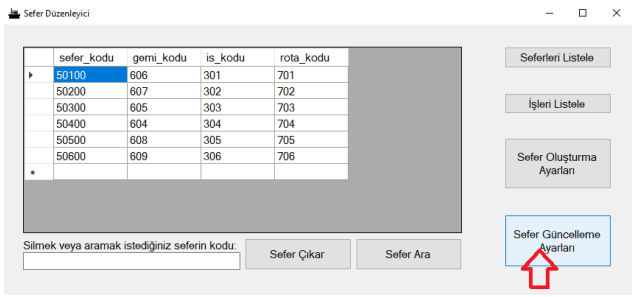


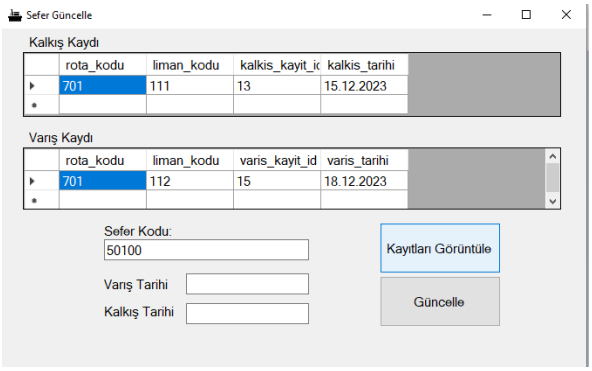
2) Sefer Ekleme

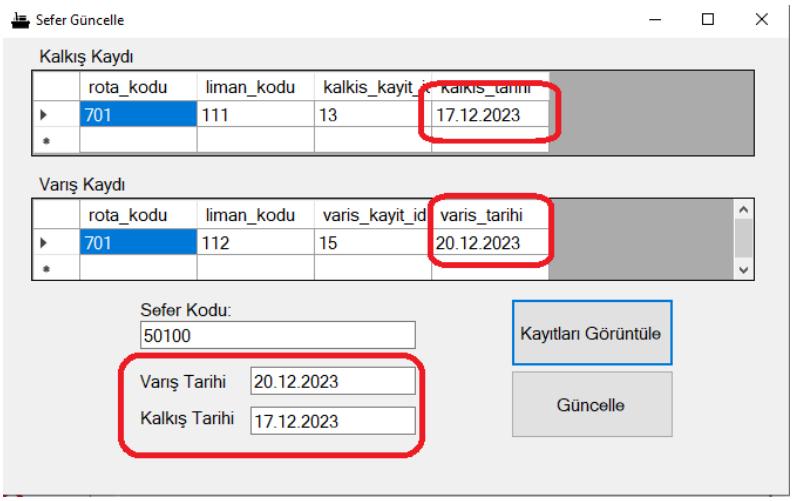




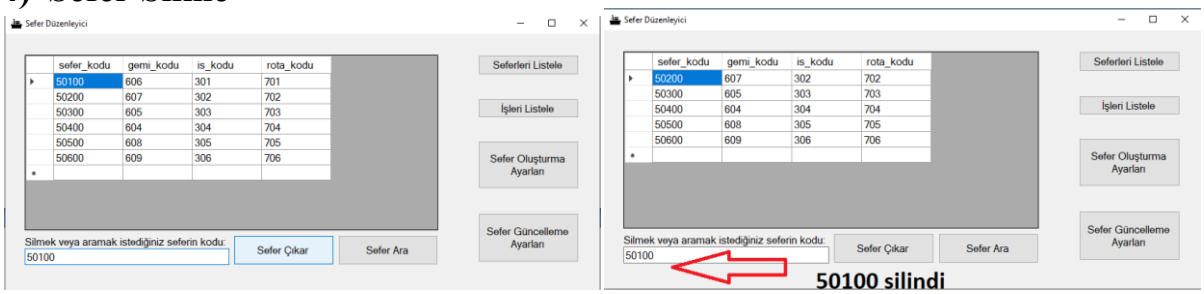
3) Sefer Kayıt Güncelleme







4) Sefer Silme



UYGULAMANIN KAYNAK KODLARI

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace VTYS_Proje
{
  internal static class Program
  {
    /// <summary>
    /// Uygulamanın ana girdi noktası.
    /// </summary>
    [STAThread]
    static void Main()
    {
      Application.EnableVisualStyles();
      Application.SetCompatibleTextRenderingDefault(false);
      Application.Run(new ana_form());
    }
  }
                                                    }
```

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace VTYS_Proje
{

  public partial class ana_form : Form
  {
    NpgsqlConnection baglanti = new NpgsqlConnection("server =localHost; port =5432; Database =proje1; user ID =postgres;
password =lE5xCn3");
    private sefer_ekle_form seferEkleF;
    private guncelleme_form guncelleme;
    public ana_form()
    {
      InitializeComponent();
      seferEkleF = new sefer_ekle_form();
      seferEkleF.FormClosed += SeferEkleFFormClosed;
```

```
    guncelleme = new guncelleme_form();
    guncelleme.FormClosed += Guncelleme_FormClosed;
}

private void Guncelleme_FormClosed(object sender, FormClosedEventArgs e)
{
    guncelleme = new guncelleme_form();
    guncelleme.FormClosed += Guncelleme_FormClosed;
}

private void Form1_Load(object sender, EventArgs e)
{

}

private void button3_Click(object sender, EventArgs e)
{
    guncelleme.Show(); // Formu göster
}

private void textBox1_TextChanged(object sender, EventArgs e)
{

}



private void sfr_listele_btn_Click(object sender, EventArgs e)
{
    string sorgu = "select*from sefer";
    NpgsqlDataAdapter da = new NpgsqlDataAdapter(sorgu, baglanti);
    DataSet ds = new DataSet();
    da.Fill(ds);
    dGv_seferler.DataSource = ds.Tables[0];
}
private void SeferEkleFFormClosed(object sender, FormClosedEventArgs e)
{
    seferEkleF = new sefer_ekle_form();
    seferEkleF.FormClosed += SeferEkleFFormClosed;
}
private void sfr_ekle_btn_Click(object sender, EventArgs e)
{
    seferEkleF.Show(); // Formu göster
}

private void dGv_seferler_CellContentClick(object sender, DataGridViewCellEventArgs e)
{

}

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

private void sfr_cikar_btn_Click(object sender, EventArgs e)
{
    baglanti.Open();
    NpgsqlCommand komt = new NpgsqlCommand("delete from sefer where sefer_kodu=@p1 ",baglanti);
    komt.Parameters.AddWithValue("@p1",int.Parse(sfr_id_txtbox.Text));
    komt.ExecuteNonQuery();
    baglanti.Close();
```

```
            MessageBox.Show("Seçilen sefer silindi","Bilgi",MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void sfr_ara_btn_Click(object sender, EventArgs e)
        {
            int sefer_idsi = int.Parse(sfr_id_txtbox.Text);
            baglanti.Open();
            NpgsqlDataAdapter da_sefr_kyt = new NpgsqlDataAdapter($"select * from sefer where sefer_kodu='{sefer_idsi}'", baglanti);
            DataTable dt1 = new DataTable();
            da_sefr_kyt.Fill(dt1);
            dGv_seferler.DataSource = dt1;
            baglanti.Close();
        }
    }
}
using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace VTYS_Proje
{
    public partial class sefer_ekle_form : Form
    {
        NpgsqlConnection baglanti = new NpgsqlConnection("server =localHost; port =5432; Database =proje1; user ID =postgres;
password =lE5xCn3");
        public sefer_ekle_form()
        {
            InitializeComponent();
        }

        private void isci_text_Click(object sender, EventArgs e)
        {

        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void is_id_label_Click(object sender, EventArgs e)
        {

        }

        private void gemiler_sekmesi_Click(object sender, EventArgs e)
        {

        }

        private void sfr_ekle_btn_Click(object sender, EventArgs e)
        {
            baglanti.Open();
            NpgsqlCommand cmd_0 = new NpgsqlCommand("insert into is_ (is_kodu,is_adi,is_suresi,sahibi_firma_id) values
(@p1,@p2,@p3,@p4)",baglanti);
            cmd_0.Parameters.AddWithValue("@p1", int.Parse(is_kodu_txtbox.Text));
            cmd_0.Parameters.AddWithValue("@p2", textBox1.Text);
            cmd_0.Parameters.AddWithValue("@p3", int.Parse(textBox2.Text));
            cmd_0.Parameters.AddWithValue("@p4", int.Parse(sirket_c_box.SelectedValue.ToString()));
```

```
        cmd_0.ExecuteNonQuery();

        NpgsqlCommand cmd_3 = new NpgsqlCommand("insert into rota (kodu,mesafe,ort_seyir_surati) values (@p1,@p2,@p3)",
baglanti);
        cmd_3.Parameters.AddWithValue("@p1", int.Parse(rota_id_textbox.Text));
        cmd_3.Parameters.AddWithValue("@p2", int.Parse(textBox5.Text));
        cmd_3.Parameters.AddWithValue("@p3", int.Parse(textBox3.Text));
        cmd_3.ExecuteNonQuery();

        NpgsqlCommand cmd_2 = new NpgsqlCommand("insert into varis_kayit (rota_kodu,liman_kodu,varis_tarihi) values
(@p1,@p2,@p3)", baglanti);
        cmd_2.Parameters.AddWithValue("@p1", int.Parse(rota_id_textbox.Text));
        cmd_2.Parameters.AddWithValue("@p2", int.Parse(varis_l_c_box.SelectedValue.ToString()));
        cmd_2.Parameters.AddWithValue("@p3", textBox6.Text);
        cmd_2.ExecuteNonQuery();

        NpgsqlCommand cmd_1 = new NpgsqlCommand("insert into kalkis_kayit (rota_kodu,liman_kodu,kalkis_tarihi) values
(@p1,@p2,@p3)", baglanti);
        cmd_1.Parameters.AddWithValue("@p1", int.Parse(rota_id_textbox.Text));
        cmd_1.Parameters.AddWithValue("@p2", int.Parse(kalkis_l_c_box.SelectedValue.ToString()));
        cmd_1.Parameters.AddWithValue("@p3", textBox7.Text);
        cmd_1.ExecuteNonQuery();

        NpgsqlCommand cmd_4 = new NpgsqlCommand("insert into sefer (sefer_kodu,gemi_kodu,is_kodu,rota_kodu) values
(@p1,@p2,@p3,@p4)", baglanti);
        cmd_4.Parameters.AddWithValue("@p1", int.Parse(sfr_id_txtbox.Text));
        cmd_4.Parameters.AddWithValue("@p2", int.Parse(gemiler_c_box.SelectedValue.ToString()));
        cmd_4.Parameters.AddWithValue("@p3", int.Parse(is_kodu_txtbox.Text));
        cmd_4.Parameters.AddWithValue("@p4", int.Parse(rota_id_textbox.Text));
        cmd_4.ExecuteNonQuery();
        baglanti.Close();
        MessageBox.Show("Sefer eklendi", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
        this.Close();
    }

    private void sefer_ekle_form_Load(object sender, EventArgs e)
    {
        baglanti.Open();
        NpgsqlDataAdapter da_gemi = new NpgsqlDataAdapter("select * from gemi",baglanti);
        DataTable dt = new DataTable();
        da_gemi.Fill(dt);
        gemiler_c_box.DisplayMember = "adi";
        gemiler_c_box.ValueMember = "gemi_kodu";
        gemiler_c_box.DataSource = dt;
        baglanti.Close();
        baglanti.Open();
        NpgsqlDataAdapter da_firma = new NpgsqlDataAdapter("select * from sirket", baglanti);
        DataTable dt_1 = new DataTable();
        da_firma.Fill(dt_1);
        sirket_c_box.DisplayMember = "sirket_adi";
        sirket_c_box.ValueMember = "sirket_id";
        sirket_c_box.DataSource = dt_1;
        baglanti.Close();
        baglanti.Open();
        NpgsqlDataAdapter da_k_liman = new NpgsqlDataAdapter("select * from liman", baglanti);
        DataTable dt_2 = new DataTable();
        da_k_liman.Fill(dt_2);
        kalkis_l_c_box.DisplayMember = "adi";
        kalkis_l_c_box.ValueMember = "kodu";
        kalkis_l_c_box.DataSource = dt_2;
        baglanti.Close();
        baglanti.Open();
        NpgsqlDataAdapter da_v_liman = new NpgsqlDataAdapter("select * from liman", baglanti);
        DataTable dt_3 = new DataTable();
        da_v_liman.Fill(dt_3);
        varis_l_c_box.DisplayMember = "adi";
```

```
            varis_l_c_box.ValueMember = "kodu";
            varis_l_c_box.DataSource = dt_3;
            baglanti.Close();

        }
        private void cmbGemi_SelectedIndexChanged(object sender, EventArgs e)
        {
            string secilenGemiAdi = gemiler_c_box.SelectedValue.ToString();


            DataTable gemiBilgileri = GetGemiBilgileri(secilenGemiAdi);
            dGv_gemi_info.DataSource = gemiBilgileri;

            DataTable gemiPersonelleri = GetGemiPersonelleri(secilenGemiAdi);
            dataGridpers.DataSource = gemiPersonelleri;
        }

        private DataTable GetGemiBilgileri(string gemiAdi)
        {
            NpgsqlDataAdapter da_gemi = new NpgsqlDataAdapter($"select * from gemi where gemi_kodu = '{gemiAdi}'", baglanti);
            DataTable dtt = new DataTable();
            da_gemi.Fill(dtt);
            return dtt;
        }

        private DataTable GetGemiPersonelleri(string gemiAdi)
        {
            NpgsqlDataAdapter da_pers = new NpgsqlDataAdapter($"select * from personel where gemi_kodu = '{gemiAdi}'", baglanti);
            DataTable dt = new DataTable();
            da_pers.Fill(dt);
            return dt;
        }

    }
}
using Npgsql;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace VTYS_Proje
{

    public partial class guncelleme_form : Form
    {

        NpgsqlConnection baglanti = new NpgsqlConnection("server =localHost; port =5432; Database =proje1; user ID =postgres;
password =lE5xCn3");
        public guncelleme_form()
        {
            InitializeComponent();
        }

        private void guncelleme_form_Load(object sender, EventArgs e)
        {

        }

        private void guncelle_btn_Click(object sender, EventArgs e)
```

```
    {
        baglanti.Open();
        NpgsqlCommand komt = new NpgsqlCommand($"UPDATE kalkis_kayit\r\nSET kalkis_tarihi = @p1 \r\nFROM sefer\r\nWHERE
kalkis_kayit.rota_kodu = sefer.rota_kodu\r\n    AND sefer.sefer_kodu = @p3;", baglanti);
        komt.Parameters.AddWithValue("@p1", kalkis_txt.Text);
        komt.Parameters.AddWithValue("@p3", int.Parse(sfr_id_txtbox.Text));
        komt.ExecuteNonQuery();
        NpgsqlCommand komt2 = new NpgsqlCommand($"UPDATE varis_kayit\r\nSET varis_tarihi = @p2 \r\nFROM sefer\r\nWHERE
varis_kayit.rota_kodu = sefer.rota_kodu\r\n    AND sefer.sefer_kodu = @p3;", baglanti);
        komt2.Parameters.AddWithValue("@p2", varis_txt.Text);
        komt2.Parameters.AddWithValue("@p3", int.Parse(sfr_id_txtbox.Text));
        komt2.ExecuteNonQuery();
        baglanti.Close();
        MessageBox.Show("Kayıtlar güncellendi", "Bilgi", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void label4_Click(object sender, EventArgs e)
    {

    }

    private void goruntule_btn_Click(object sender, EventArgs e)
    {
        int sefer_idsi = int.Parse(sfr_id_txtbox.Text);
        baglanti.Open();
        NpgsqlDataAdapter da_sefr_kyt = new NpgsqlDataAdapter($"SELECT kk.*\r\nFROM kalkis_kayit kk\r\nINNER JOIN sefer s ON
kk.rota_kodu = s.rota_kodu\r\nWHERE s.sefer_kodu = '{sefer_idsi}';", baglanti);
        DataTable dt1 = new DataTable();
        da_sefr_kyt.Fill(dt1);
        dataGridKayit.DataSource = dt1;
        NpgsqlDataAdapter da_sefr_vars = new NpgsqlDataAdapter($"SELECT vk.*\r\nFROM varis_kayit vk\r\nINNER JOIN sefer s ON
vk.rota_kodu = s.rota_kodu\r\nWHERE s.sefer_kodu = '{sefer_idsi}';", baglanti);
        DataTable dt = new DataTable();
        da_sefr_vars.Fill(dt);
        dataGridVaris.DataSource = dt; baglanti.Close();
    }
  }
}
```