Veritabanı Yönetim Sistemleri Raporu

Bir seyahat uygulaması yapmayı plandık. Kullanıcıların gezdiği yerleri puanlamasıyla favori yerler belirlenir.

Kısaca iş kuralları:

1- Kullanicilar; admin,moderator ya da turist kullanıcı olmalıdır.

2- Kullanıcılar birden fazla yorum yapabilir ama 1 yorum yalnızca 1 kişiye ait olmalıdır.

3- Kullanıcılar en az bir en fazla çok etkinliğe katılabilir.

4- Kullanıcıların en az bir en fazla çok favori yeri olabilir.

5- Etkinlik katılımcıları en az bir en çok bir etkinliğe katılabilir.

6- Bir ülkede en az bir en fazla çok sayıda şehir bulunabilir.

7- Bir şehirde en az bir en fazla çok sayıda otel ve restoran bulunabilir. Oteller adres ve telefon no bilgilerine sahiptir .Restoranlarda mutfak türü bilgisi yer alır.

8- Bir restoranda bir mutfak türü vardır fakat bir mutfak türü birden fazla restoranda yer alabilir.

9- Bir hava durumu birden fazla şehirde olabilir ama bir şehirde aynı anda birden fazla hava durumu olamaz.

10- Bir şehirde birden fazla etkinlik olabilir ama bir etkinlik bir şehirde olmalıdır.

11- Bir şehirde birden fazla gezilecek yerler olabilir ama bir gezilecek yer en fazla bir şehirde bulunur.

12- Gezilecek yerlerde birden fazla favori yer bulunabilir ama bir favori yer yalnızca bir gezilecek yerde bulunur. Ücret bilgisini de içerir.

13- Taşıma araçları taşıma türünü belirtir.

14- Etkinlik katılımcıları katılımcı adlarını ,etkinlik id'sini, kullanıcı id'sini ve katılımcı id'sini tutar.

İlişkisel Model :

Kullanicilar(KullaniciId:int,KullaniciAdi:varchar,Email,text,KayitTarihi:date,AdminRol:varchar,ModeratorRol:varchar,TuristRol:varchar,Yas:int)

AdminKullanicilar(KullaniciId:int,AdminRol:varchar)

ModeratorKullanicilar(KullaniciId:int,ModeratorRol:varchar)

TuristKullanicilar(KullaniciId:int,AdminRol:varchar)

Oteller(OtelId:int,OtelAdi:varchar,SehirId:int,Adres:text,TelefonNo:text)

Ulkeler(Ulkeid:int,UlkeAdi:varchar,Baskent:varchar,Nufus:int,ParaBirimi:varchar,SehirSayisi:int)

Yorumlar(Yorumid:int,KullaniciId:int,Yerid:int,Yorum:text,Puan:int)

HavaDurumu(HavaDurumuId:int,Sicaklik:int,SehirId:int,HavaDurumuAd:varchar)

Sehirler(SehirId:int,SehirAdi:varchar,UlkeId:int,Nufus:int)

GezilecekYerler(YerId:int,YerAdi:varchar,SehirId:int,GirisUcreti:int)

FavoriYerler(FavoriId:int,KullaniciId:int,YerId:int)

EtkinlikKatilimcilari(KatilimciId:int,EtkinlikId:int,KullaniciId:int,KatilimciAdi:varchar)

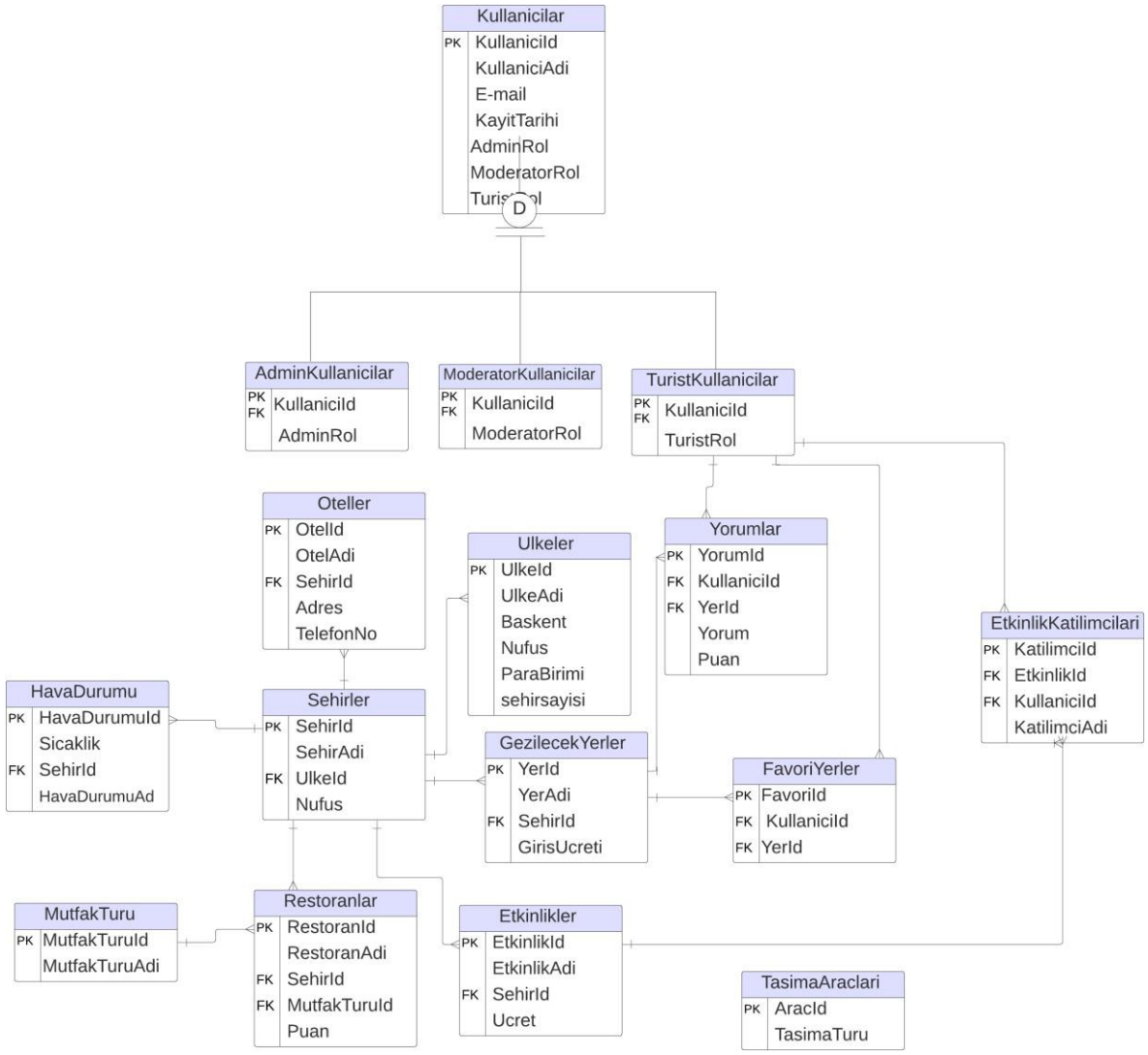
MutfakTuru(MutfakTuruId:int,MutfakTuruAdi:varchar)

Restoranlar(RestoranId:int,RestoranAdi.varchar,SehirId:int,MutfakTuruId:int,Puan:int)

Etkinlikler(EtkinlikId:int,EtkinlikAdi:varchar,SehirId:int,Ucret:int)

TasimaAraçlari(AracId:int,TasimaTuru:varchar)

VARLIK BAĞINTI DİYAGRAMI:



SQL İfadeleri :

CREATE TABLE “public”.”Kullanicilar”(

“KullaniciId” INTEGER,

```
“KullaniciAdi” VARCHAR (30),
“Email” TEXT,
“KayitTarihi” DATE,
“ModeratorRol” VARCHAR(30),
“AdminRol” VARCHAR(30),
“TuristRol” VARCHAR (30),
“Yas” INTEGER ,
CONSTRAINT “KullanicilarPK” PRIMARY KEY (“KullaniciId”));
CREATE TABLE “public”.”AdminKullanicilar”(
“KullaniciId” INTEGER,
“AdminRol” VARCHAR(30),
CONSTRAINT “AdminKullanicilarPK” PRIMARY KEY (“KullaniciId”),
CONSTRAINT “AdminKullanicilarFK” FOREIGN KEY (“KullaniciId”)
REFERENCES “public”.”Kullanicilar”(“KullaniciId”)
    ON DELETE CASCADE ON UPDATE CASCADE
);
CREATE TABLE “public”.”ModeratorKullanicilar”(
“KullaniciId” INTEGER,
“ModeratorRol” VARCHAR(30),
CONSTRAINT “ModeratorKullanicilarPK” PRIMARY KEY (“KullaniciId”),
CONSTRAINT “ModeratorKullanicilarFK” FOREIGN KEY (“KullaniciId”)
REFERENCES “public”.”Kullanicilar”(“KullaniciId”)
    ON DELETE CASCADE ON UPDATE CASCADE
);
```

```
CREATE TABLE “public”.”TuristKullanicilar”(
“KullaniciId” INTEGER,
“TuristRol” VARCHAR(30),
CONSTRAINT “TuristKullanicilarPK” PRIMARY KEY (“KullaniciId”),
CONSTRAINT “TuristKullanicilarFK” FOREIGN KEY (“KullaniciId”)
REFERENCES “public”.”Kullanicilar”(“KullaniciId”)
        ON DELETE CASCADE ON UPDATE CASCADE
);



select "public"."Kullanicilar"."KullaniciAdi",
        "public"."Kullanicilar"."Email",
        "public"."Kullanicilar"."KayitTarihi",
        "public"."Kullanicilar"."ModeratorRol",
        "public"."Kullanicilar"."AdminRol",
        "public"."Kullanicilar"."TuristRol"
 from "public"."Kullanicilar"
 left outer join "public"."AdminKullanicilar" on
"public"."Kullanicilar"."KullaniciId"="public"."AdminKullanicilar"."Kull
aniciId"
 left outer join "public"."TuristKullanicilar" on
"public"."Kullanicilar"."KullaniciId"="public"."TuristKullanicilar"."Kulla
niciId"
 left outer join "public"."ModeratorKullanicilar" on
"public"."Kullanicilar"."KullaniciId"="public"."ModeratorKullanicilar"."
KullaniciId"
```

FONKSİYONLAR

1) **Her ülkenin nüfus yoğunluğunu hesaplayan fonksiyon. (calculate_population_density):**

```
CREATE OR REPLACE FUNCTION calculate_population_density(p_ulke_id INTEGER)
RETURNS NUMERIC AS $$
DECLARE
   total_population INTEGER;
   city_count INTEGER;
   population_density NUMERIC;
BEGIN
   -- Belirli bir ülkenin toplam nüfusunu al
   SELECT nufus INTO total_population
   FROM ulkeler
   WHERE ulke_id = p_ulke_id;

   -- Belirli bir ülkenin şehir sayısını al
   SELECT sehirsayisi INTO city_count
   FROM ulkeler
   WHERE ulke_id = p_ulke_id;

   -- Nüfus yoğunluğunu hesapla
   IF city_count > 0 THEN
      population_density := total_population / city_count;
   ELSE
      population_density := 0;
   END IF;

   RETURN population_density;
END;
$$ LANGUAGE plpgsql;
```

Denemek için: SELECT calculate_population_density(ulkeid);

2) **Bir şehrin yurt içinde mi yurt dışında mı olduğunu gösteren fonksiyon. (check_yurtici_yurtdisi):**

```
CREATE OR REPLACE FUNCTION check_yurtici_yurtdisi(p_sehir_id INTEGER)
RETURNS VARCHAR(50) AS $$
DECLARE
   ulke_id_temp INTEGER;
```

```
    yurtici_yurtdisi VARCHAR(50);
BEGIN
    -- Verilen sehir_id'ye ait ulke_id'yi al
    SELECT ulke_id INTO ulke_id_temp FROM sehirler WHERE sehir_id = p_sehir_id;

    -- Yurt içi veya yurt dışı kontrolü
    IF ulke_id_temp IS NOT NULL THEN
        IF ulke_id_temp = 3 THEN
            yurtici_yurtdisi := 'Yurt İçi';
        ELSE
            yurtici_yurtdisi := 'Yurt Dışı';
        END IF;
    ELSE
        yurtici_yurtdisi := 'Geçersiz Sehir ID';
    END IF;

    RETURN yurtici_yurtdisi;
END;
$$ LANGUAGE plpgsql;

Denemek için: SELECT sehirad, check_yurtici_yurtdisi(sehirid) as yurtici_yurtdisi
FROM sehirler;
```

3) Bir puanın 1 ile 10 arasında olması gerektiğinin kontrolünü yapan fonksiyon. (puan_kontrol):

```
CREATE OR REPLACE FUNCTION puan_kontrol(p_kullanici_id INTEGER, p_yer_id
INTEGER, p_puan INTEGER)

RETURNS VOID AS $$

BEGIN

  IF p_puan < 1 OR p_puan > 10 THEN

    RAISE EXCEPTION 'Geçersiz puan. Puan 1 ile 10 arasında olmalıdır.';

  ELSE

    INSERT INTO "Yer_Puanlar" ("KullaniciId", "YerId", "Puan")

    VALUES (p_kullanici_id, p_yer_id, p_puan);

  END IF;

END;

$$ LANGUAGE plpgsql;
```

4) Bir yorumun en az 10 karakterden oluşması gerektiğini kontrol eden fonksiyon. (yorum_kontrol):

```
CREATE OR REPLACE FUNCTION  yorum_kontrol(p_kullanici_id INTEGER, p_yer_id
INTEGER, p_yorum_metni TEXT, p_puan INTEGER)
RETURNS VOID AS $$
DECLARE
   min_yorum_uzunlugu INTEGER := 10; -- Minimum yorum uzunluğu
BEGIN
   IF LENGTH(p_yorum_metni) < min_yorum_uzunlugu THEN
      -- Yorum çok kısa ise burada yapılacak işlemleri ekleyebilirsiniz.
      RAISE EXCEPTION 'Yorum çok kısa. Minimum uzunluk: %', min_yorum_uzunlugu;
   ELSE
      -- Yorumu ekleyebilir veya başka işlemler yapabilirsiniz.
      INSERT INTO "Yorumlar" ("kullaniciId", "yerid", "yorum", "puan")
      VALUES (p_kullanici_id, p_yer_id, p_yorum_metni, p_puan);
   END IF;
END;
$$ LANGUAGE plpgsql;

Denemek için: DO $$
BEGIN
   PERFORM yorum_kontrol(18, 'çok kısa', 2, 130, 2);
END $$;
```

TRIGGERLAR

1) Kullanıcının emailinin domain kısmını kontrol eden fonksiyon. (chech_turistkullanici_email) :

```
CREATE OR REPLACE FUNCTION check_turistkullanici_email()
RETURNS TRIGGER AS $$
BEGIN
   IF NEW."KullaniciTipi" = 'admin' AND POSITION('@gmail.com' IN NEW."Email")
= 0 THEN
      RAISE EXCEPTION 'Turist kullanıcıları için e-posta adresi @gmail.com domaini
ile bitmelidir.';
   END IF;

   RETURN NEW;
END;
```

```
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_turistkullanicilar_email_trigger
BEFORE INSERT ON “public”."Kullanicilar"
FOR EACH ROW EXECUTE FUNCTION check_turistkullanicilar_email();
```

2) Silinen kullanıcıyı diğer tablolardan da silen trigger (delete_user_data):

```
CREATE OR REPLACE FUNCTION delete_user_data()
RETURNS TRIGGER AS $$
BEGIN
   DELETE FROM "Moderator_Kullanicilar" WHERE "KullaniciId" =
OLD."KullaniciId";
   DELETE FROM "Admin_Kullanicilar" WHERE "KullaniciId" = OLD."KullaniciId";
   DELETE FROM "Turist_Kullanicilar" WHERE "KullaniciId" = OLD."KullaniciId";

   RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER delete_user_data_trigger
AFTER DELETE ON “public”."Kullanicilar"
FOR EACH ROW EXECUTE FUNCTION delete_user_data();
```

Denemek için: DELETE FROM "Kullanicilar" WHERE "KullaniciId" = 1;

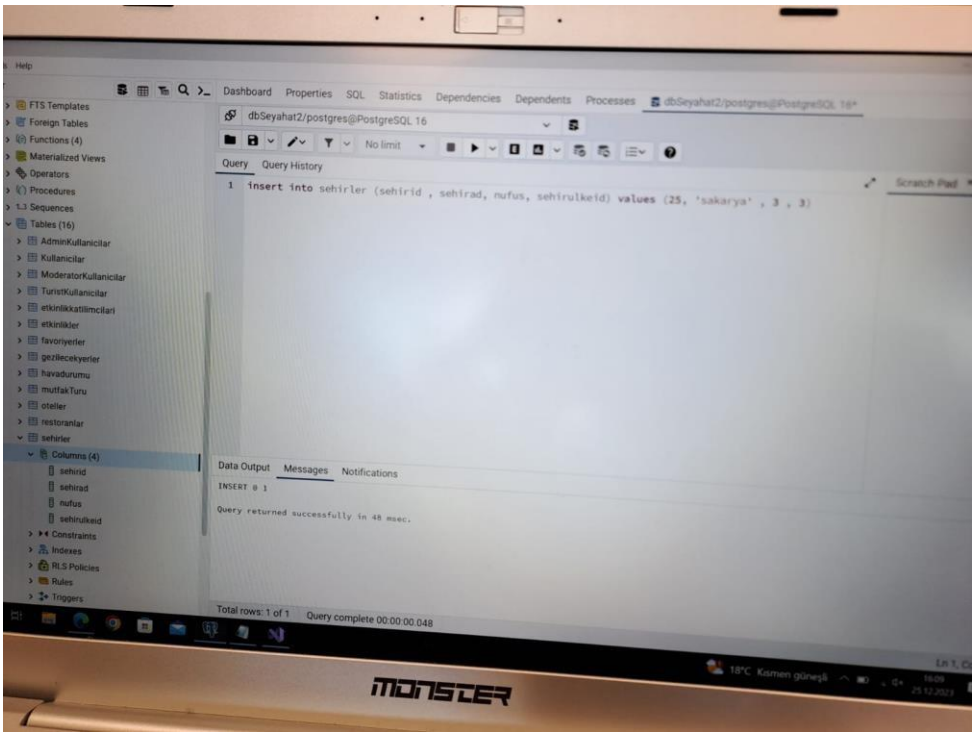
3) Yeni kullanıcı eklenince ekrana hoş geldin mesajı yazan trigger. (hosgeldin_mesaji):

```
CREATE OR REPLACE FUNCTION hosgeldin_mesaji()
RETURNS TRIGGER AS $$
BEGIN
   -- Yeni kullanıcı eklenince hoşgeldiniz mesajı yaz
   RAISE NOTICE 'Hoş geldiniz, %! Kullanıcı eklendi!', NEW.kullaniciadi;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER hosgeldin_mesaji_trigger
```

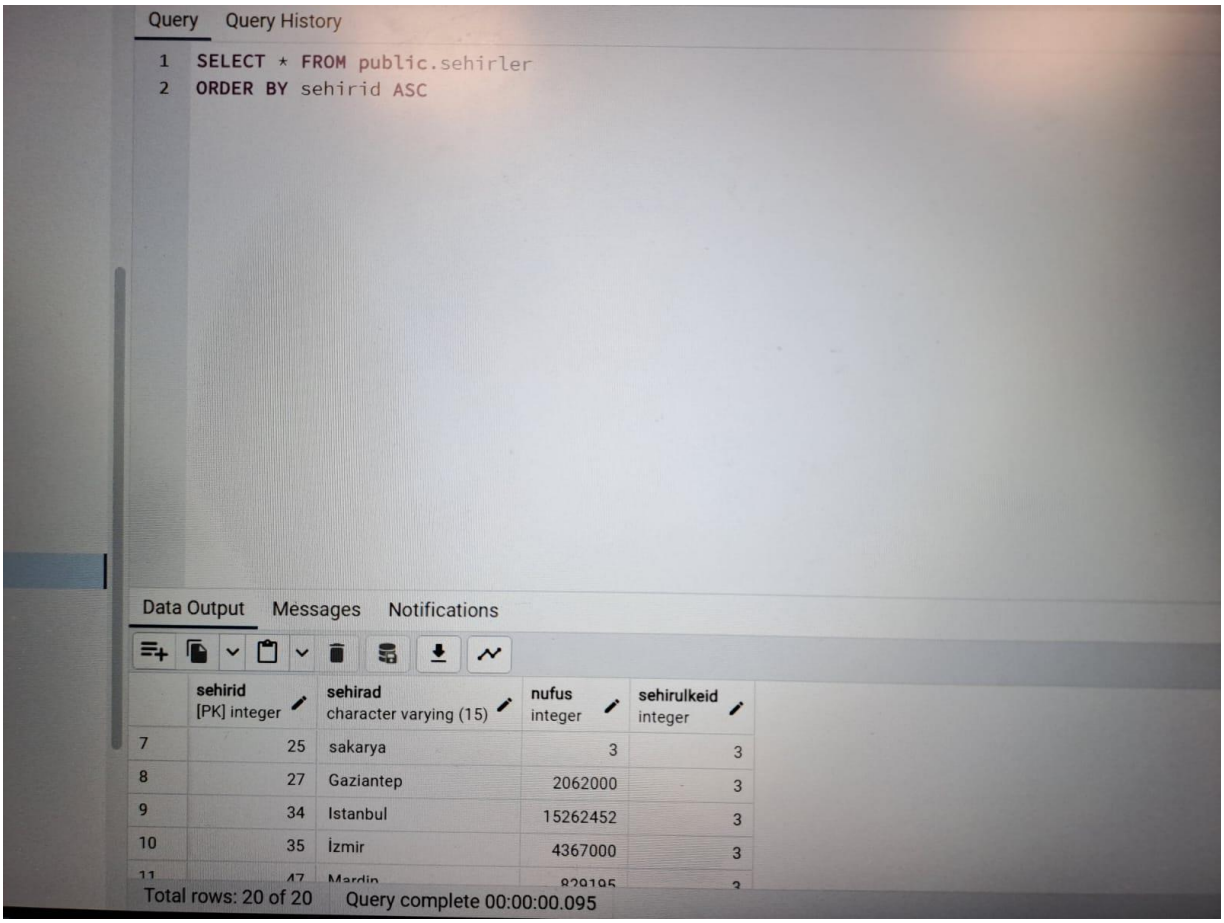
```
AFTER INSERT ON “public”."Kullanicilar"
FOR EACH ROW
EXECUTE FUNCTION hosgeldin_mesaji();
```

4) Yeni kullanıcı eklenirken 18 yaş sınırı koyan, 18 yaşından küçükleri eklemeyen trigger. (yas_kontrol):

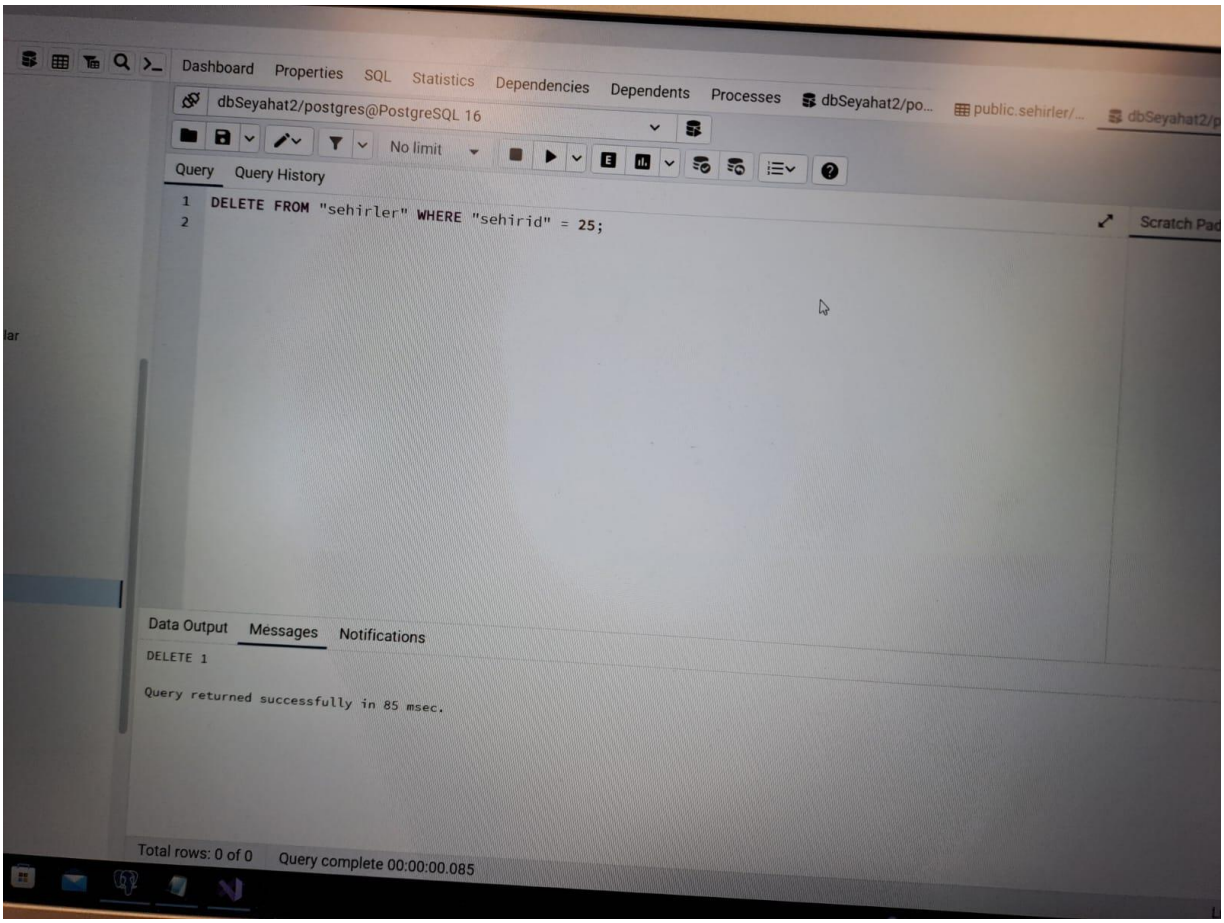
```
CREATE OR REPLACE FUNCTION yas_kontrol()
RETURNS TRIGGER AS $$
BEGIN
   -- Kullanıcının yaşını kontrol et (örneğin, yaş sınırı 18)
   IF COALESCE(NEW.yas, 0) < 18 THEN
      RAISE EXCEPTION 'Kullanıcı (%s) yaş kontrolü başarısız oldu. Yas: %s',
NEW.kullaniciadi, NEW.yas;
   END IF;

   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER yas_kontrol_trigger
BEFORE INSERT ON kullanicilar
FOR EACH ROW
EXECUTE FUNCTION yas_kontrol();
```


Help
FTS Templates
Foreign Tables
Functions (4)
Materialized Views
Operators
Procedures
Sequences
Tables (16)
AdminKullanicilar
Kullanicilar
ModeratorKullanicilar
TuristKullanicilar
etkinlikkatilimcilari
etkinlikler
favoriyerler
gezilecekyerler
havadurumu
mutfakTuru
oteller
restoranlar
sehirler
Columns (4)
sehirid
sehirad
nufus
sehirulkeid
Constraints
Indexes
RLS Policies
Rules
Triggers
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
dbSeyahat2/postgres@PostgreSQL 16
No limit
Query
Query History
1 insert into sehirler (sehirid , sehirad, nufus, sehirulkeid) values (25, 'sakarya' , 3 , 3)
Scratch Pad
Data Output
Messages
Notifications
INSERT 0 1
Query returned successfully in 48 msec.
Total rows: 1 of 1
Query complete 00:00:00.048
18°C Kismen güneşli
MONSTER

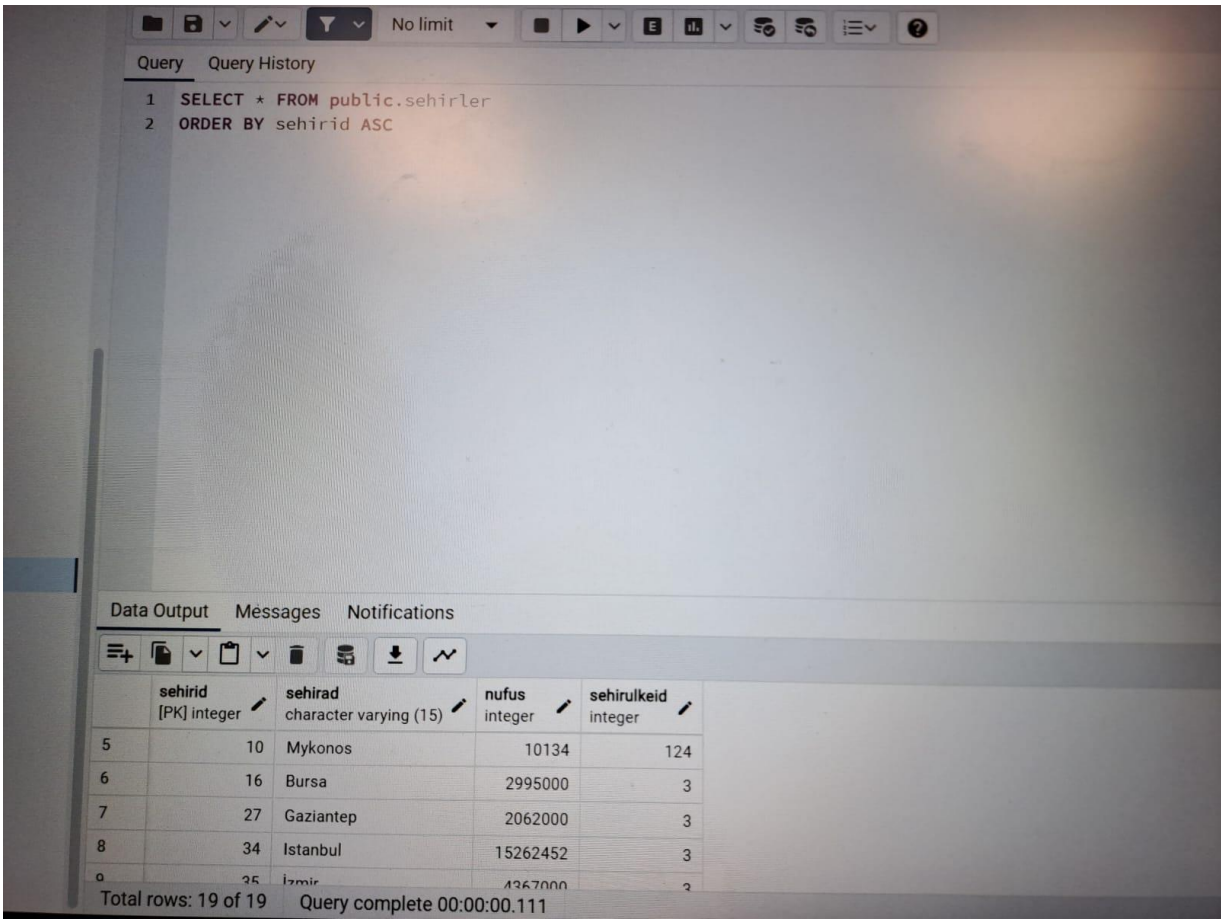
Ekleme



Eklenmiş hali


Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
dbSeyahat2/postgres@PostgreSQL 16
No limit
Query
Query History
1 DELETE FROM "sehirler" WHERE "sehirid" = 25;
2
Scratch Pad
Data Output
Messages
Notifications
DELETE 1
Query returned successfully in 85 msec.
Total rows: 0 of 0
Query complete 00:00:00.085

Silme



Silinmiş hali.