BLM3041 Veri Tabanı Yönetimi Dönem Projesi Raporu

Emin Kürşat Doğan | 21011033

Aziz Çifçibaşı | 21011104

Ali Eren Arık | 21011050

Kemal Ata Türkoğlu | 2101160

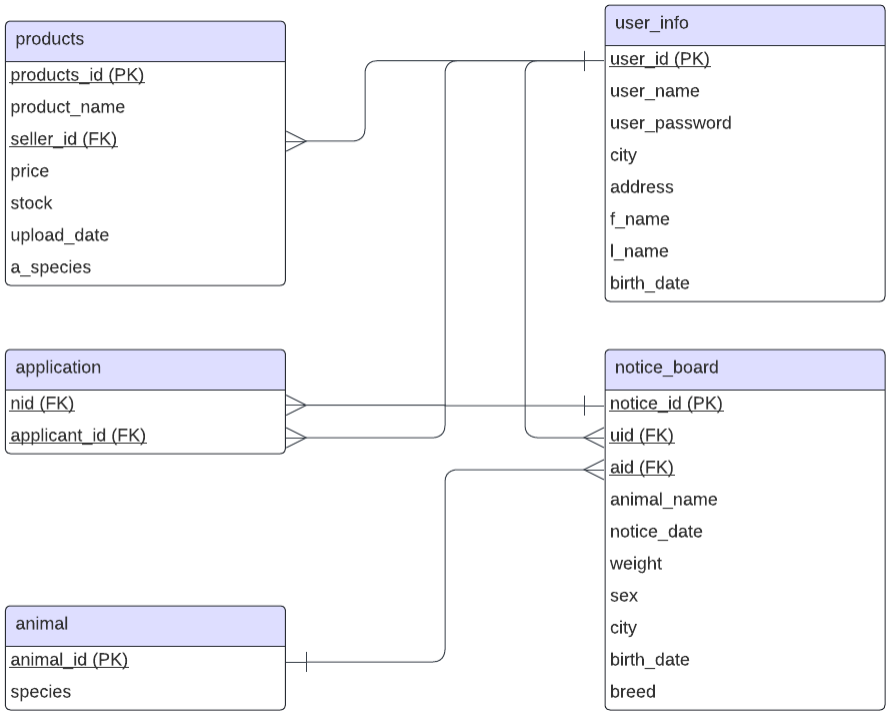
Dersin Yürütücüsü: Dr. Öğr. Üyesi Mustafa Utku KALAY

Proje Konusu: PetSet- Hayvan Sahiplenme Sistemi

Kullanıcı, sisteme kullanıcı adı ve şifre ile kaydolup, login ekranından giriş yapar. Seçtiği hayvan türüne göre (kedi, köpek, kuş vb.) sahiplenme ilanları listelenir. Kullanıcı listelenen ilanlara başvuruda bulunur. Her bir sahiplenme ilanına kimlerin hangi şehirlerden başvurduğu listelenmelidir. Kullanıcı başvurduktan sonra onun da bulunduğu şehirle birlikte başvuranlar arasında ismi görünmelidir. Kullanıcı kendi sayfasında başvurduğu ilanları görüntüleyebilmeli, adres vb. kişisel bilgilerini girip güncelleyebilmelidir. Ek özellikler olarak; kullanıcı kendi evcil hayvanını da sahiplenme ilanına koyabilir. Alışveriş modülünden çeşitli evcil hayvan ürünlerinin satın alımı yapılabilir.

ER DİYAGRAMI

Proje'de bulunan veri tabanının ER Diyagramı:



Projemiz insanların hem evcil hayvan hem evcil hayvan ürünü alıp satabildikleri bir sahiplendirme sitesi için veri tabanı ve sorgu sistemidir.

TABLOLAR

User_info:

Öncelikle her bir ayrı kullanıcının kendi alanında siteyi kullanabilmesi için user_info tablosu oluşturulur. User_id Unique bir Primary Key olup her kullanıcı için farklı iken diğer bilgiler kullanıcıdan input olarak alınır ve sisteme kaydedilir. Diğer tabloların çoğu Foreign Key'lerine referans olarak alır. Onun dışında alacağımız input bilgiler: kullanıcı adı (user_name), şifre (user_password), şehir (city), adres (address), isim ve soyismi (f_name, l_name), ve doğum günüdür (birth_date). Doğum gününe göre yaşı 18'den küçükler siteyi kullanamaz. Kullanıcılar ilan açarken şehir bilgileri otomatik olarak kullanıcı bilgilerinden çekilir.

Animal:

Hayvan türleri için açılan Animal tablosunda 2 veri tipi vardır: hayvanın id'si (animal_id) ve türü (species). Bu tablo user tarafından erişilmez ve içinde sahiplendirme sistemine koyulabilecek hayvan türlerini tutar. Admin tarafından SQL kodu tarafında bu listeye yeni hayvanlar eklenebilir, Primary Key olan animal_id her ekleme için artar, species de Unique olduğu için aynı tür iki kere eklenemez. Bu türlere örneğin kedi, köpek, kuş gibi örnekler verebiliriz. Bir kedinin turuncu mu yoksa tekir mi olduğu gibi bilgiler burada yer almaz. Kullanıcıların sahiplendirme ilanı açacağı veya arama yapacağı zaman önlerine açılan liste bu tablodaki unique species özelliğini listeler, bu sayede kullanıcıların seçebileceği tür seçimi sadece bu tablodaki hayvanlardan olabilir.

Notice_board:

Hayvan sahiplendirme ilanlarının bulunduğu tablodur. İlandaki hayvana ait isim, kütle, cinsiyet, cins, doğum tarihi ve tür id bilgileri bulunurken; ilanın verildiği tarih, ilanı veren kullanıcının id bilgisi ve bu ilana ait bir id bu tabloda

tutulur. İlana ait id bilgisini tutan notice_id bu tablo için primary keydir ve bu değer her girilen ilan trigger tetikler ve otomatik olarak artarak tablonun bu sütununa atanır. uid ve aid sütunu bu tablo için foreign keydir. Uid User_info, aid animal_id tablosunu referans alacak şekilde tasarlanmıştır. Sisteme giren kullanıcı ilan verirken animal_name, weight, sex, breed ve birth_day sütunları için bilgileri girerek, hayvanın türünü ise aid sütununun işaret ettiği animal tablosundaki türlerin listelendiği listeden seçer. Uid ve city değerleri ise sisteme bağlanmış kullanıcı ile aynı olacak şekilde tasarlanmıştır. Ayrıca birth_day sütunu gelecekten bir tarih seçilip mantıksal bir hata oluşmaması için birth_day < current() kısıtı kullanılmıştır.

Application:

Verilen ilanlara başvuruların tutulduğu tablodur. Hangi ilana başvuru yapıldığını tutan nid değeri notice_board tablosuna referans veren bir foreign keydir. Başvurunun kimin tarafından yapıldığını tutan applicant_id sütunu ise user_info tablosuna referans veren bir foreign keydir. Sisteme giriş yapan kullanıcı bir ilana başvurduğu zaman bu ilanın idsi olan notice_id, nid değeri olarak; kullanıcının idsi olan user_id, applicant_id değeri olarak bu sütunlara eklenir.

Products:

Hayvanlar ile ilgili her türlü eşyanın alınıp satılabilmesi için gerekli bilgilerin tutulduğu tablodur. Ürünler bu tabloda ürün adı (product_name), satıcı id (seller_id), fiyat (price), stoktaki ürünlerin sayısı (stock), ürün ilanının verildiği tarih (upload_date), ve bu ürünün hangi hayvan türü için verildiğini saklayan (a_species) sütunlardaki bilgiler ile tutulur. Her bir ürün ilanı için bu ilan eklendiği zaman tetiklenen trigger ile otomatik olarak atanan bir değer olan product_id bu tablo için primary keydir. Ürünün satıcısını tutan seller_id ise user_info tablosunu referans alan bir foreign keydir.

FONKSİYONLAR VE TRIGGERLAR

Bu bölümde SQL kısmında tanımladığımız fonksiyonlar ve triggerların ne yaptığı bilgisi verilmektedir.

searchAll: Sisteme giren kullanıcının başvurmadığı ve ilanı kendisi vermediği tüm ilanları tumtip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak sisteme girmiş olan kullanıcının idsini alır.

searchBySpecies: Sisteme giren kullanıcının seçtiği belli bir hayvan türüne göre kullanıcının başvurmadığı ve kullanıcı tarafından verilmemiş tüm ilanları tumtip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak sisteme girmiş olan kullanıcının idsini ve seçilmiş olan türü alır.

searchByTown: Sisteme giren kullanıcının seçtiği belli bir şehirde verilmiş, kullanıcının başvurmadığı ve kullanıcı tarafından verilmemiş tüm ilanları tumtip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak sisteme girmiş olan kullanıcının idsini ve seçilmiş olan şehri alır.

searchBySpeciesAndTown: Sisteme giren kullanıcının seçtiği belli bir şehirde verilmiş olan seçili hayvan türüne göre kullanıcının başvurmadığı ve kullanıcı tarafından verilmemiş tüm ilanları tumtip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak sisteme girmiş olan kullanıcının idsini, seçilmiş olan şehri ve hayvan türünü alır.

listMyNotices: Sisteme giren kullanıcının vermiş olduğu tüm sahiplendirme ilanlarını mynoticetip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak sisteme girmiş olan kullanıcının idsini alır.

ApplicantList: Sisteme giren kullanıcının vermiş olduğu bir sahiplendirme ilanına başvuran kullanıcıları myapplicanttip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak ilan idsini alır.

listAllProducts: Sistemde satılmakta olan tüm ürünlerin listelendiği ve protip tipinde bir query döndüren sql fonksiyondur. Girdi almamaktadır.

searchProductByAnimal: Sistemde satılmakta olan ürünlerin seçili hayvan türüne göre listelendiği ve protip tipinde bir query döndüren sql fonksiyondur. Girdi olarak seçili hayvanın idsini alır.

listMyApplications: Sisteme giren kullanıcının başvurduğu ilanları mynoticetip type'ında bir query olarak döndüren sql fonksiyonudur. Bu fonksiyon girdi olarak sisteme girmiş olan kullanıcının idsini alır.

trig_fonk_add_appl: Bu trigger fonksiyonu kullanıcının bir ilana birden fazla başvuru yapamaması için oluşturulan bir fonksiyondur. Trigger application tablosu üzerinde tanımlı olup, bu tabloya girdi girilmeden önce tetiklenir.

trig_fonk_add_uid: Bu trigger fonksiyonu kullanıcı kaydolduğunda kullanıcının idsini otomatik olarak use_id sequence'ına göre arttırarak belirleyen bir fonksiyondur. Trigger user_info tablosu üzerinde tanımlı olup, bu tabloya girdi girilmeden önce tetiklenir.

trig_fonk_add_aid: Bu trigger fonksiyonu yeni bir hayvan türü kaydedildiği zaman hayvan idsini otomatik olarak ani_id sequence'ına göre arttırarak belirleyen bir fonksiyondur. Trigger animal tablosu üzerinde tanımlı olup, bu tabloya girdi girilmeden önce tetiklenir.

trig_fonk_add_nid: Bu trigger fonksiyonu kullanıcı ilan verdiğinde ilanın idsini otomatik olarak not_id sequence'ına göre arttırarak belirleyen bir fonksiyondur. Trigger notice_board tablosu üzerinde tanımlı olup, bu tabloya girdi girilmeden önce tetiklenir.

trig_fonk_add_pid: Bu trigger fonksiyonu yeni bir ürün kaydedildiği zaman ürün idsini otomatik olarak pro_id sequence'ına göre arttırarak belirleyen bir fonksiyondur. Trigger products tablosu üzerinde tanımlı olup, bu tabloya girdi girilmeden önce tetiklenir.

trig_fonk_delete_pid: Bu trigger fonksiyonu products tablosundan stok sayısına göre ürün alımı gerçeklendikten sonra silinmesi veya stoğun bir azaltılmasını sağlayan bir fonksiyondur. Trigger products tablosu üzerinde tanımlı olup, bu tablodan girdi silinmeden önce tetiklenir.

trig_fonk_delete_notice_applicants: Bu trigger fonksiyonu bir ilan silindiğinde bu ilana başvuruları application tablosundan silen bir fonksiyondur. Trigger notice_board tablosu üzerinde tanımlı olup, bu tablodan girdi silinmeden önce tetiklenir.

Proje İçin İstenen Özellikler

1. ***Oluşturacağınız veritabanı en az 4 tablo içermelidir. Her tabloda en az 10 kayıt bulunmalıdır.***
   Veritabanımız “user_info”, “animal”, “products”, “notice_board” ve “application” tablolarından oluşur. Bu tabloların her birinde en az10 adet kayıt bulunur.
2. ***Tablolarınızda primary key ve foreign key kısıtlarını kullanmalısınız.***
   Her bir tablo oluşturulurken primary key ve foreign key kısıtları ile oluşturulur.
3. ***En az bir tabloda silme kısıtı ve sayı kısıtı olmalıdır.***
   User_info tablosunda 18 yaşından küçük bir kişinin kaydedilmesini engelleyen bir sayı kısıtı:
4. ***Arayüzden en az birer tane insert, update ve delete işlemi gerçekleştirilebilmelidir.***
   Arayüzden çağrılan insert, update ve delete işlemleri aşağıdadır:

```
public void MainAdoptAnimal(int noticeID)
{
    DBManager.dbManager.ExecuteQuery(0, DBManager.dbConnection, "INSERT INTO application VALUES(@param1,@param2)", noticeID,UserID);
    ToggleChanged();
}
```

```
 }
 DBManager.dbManager.ExecuteQuery(0, DBManager.dbConnection, "DELETE FROM products WHERE product_id = @param1", id);
 MM_ProductMarketButton();
```

```
DBManager.dbManager.ExecuteQuery(2, DBManager.dbConnection,
"UPDATE user_info SET user_password = @param1, city = @param2, address = @param3, " +
"f_name = @param4, l_name = @param5 WHERE user_name = @param6"
, UM_passwordIField.text, UM_cityIField.text, UM_addressIField.text
```

5. ***Arayüzden girilecek bir değere göre ekrana sonuçların listelendiği bir sorgu yazmalısınız.***
   Ara yüzden seçilen hayvan türüne göre sonuç döndüren searchBySpecies fonksiyonu:

```
CREATE OR REPLACE FUNCTION searchBySpecies(tur VARCHAR(20), myID INTEGER)
RETURNS SETOF tumtip AS $$
DECLARE
    ilan tumtip;
BEGIN
    RETURN QUERY
        SELECT n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.anim
        FROM notice_board n
        JOIN animal a ON n.aid = a.animal_id
        JOIN user_info u ON n.uid = u.user_id
        WHERE a.species = tur AND myID <> u.user_id
        AND NOT EXISTS (
            SELECT 1
            FROM application app
            WHERE app.nid = n.notice_id AND app.applicant_id = myID
        );
END;
$$ LANGUAGE 'plpgsql';
```

6. ***Arayüzden çağrılan sorgulardan en az biri "view" olarak tanımlanmış olmalıdır.***

   Applicant_info view'ı arayüzden çağırılır ve ad, soyad, kullanıcı id ve adres bilgilerini yazdıran bir viewdır:

```
CREATE VIEW applicant_info AS
    SELECT f_name, l_name, user_id, address
    FROM user_info;
```

7. ***En az bir adet "sequence" oluşturmalı ve arayüzden yapılacak insert sırasında ilgili sütundaki değerlerin otomatik olarak atanmasını sağlamalısınız.***

   use_id sequence'ı user_info tablosuna girdi eklendiğinde tetiklenen trig_fonk_add_uid fonksiyonu ile istenen koşul sağlanmaktadır.

```
CREATE SEQUENCE ani_id MINVALUE 1 MAXVALUE 20 INCREMENT BY 1;
CREATE SEQUENCE use_id MINVALUE 100 MAXVALUE 120 INCREMENT BY 1;
CREATE SEQUENCE not_id MINVALUE 200 MAXVALUE 220 INCREMENT BY 1;
CREATE SEQUENCE pro_id MINVALUE 300 MAXVALUE 320 INCREMENT BY 1;
```

```
-- New User's ID: Sequence Increaser
CREATE OR REPLACE FUNCTION trig_fonk_add_uid()
RETURNS TRIGGER AS $$
BEGIN
    IF (NEW.user_id = 0) THEN
        NEW.user_id := NEXTVAL('use_id');
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE 'plpgsql';
---------------------------------------
CREATE TRIGGER add_uid
BEFORE INSERT
ON user_info
FOR EACH ROW
EXECUTE FUNCTION trig_fonk_add_uid();
```

8. ***Arayüzden çağrılan sorgulardan en az birinde union veya intersect veya except kullanmış olmalısınız.***

   Aşağıdaki SearchAll fonksiyonu except kullanmaktadır:

```
CREATE OR REPLACE FUNCTION searchAll(myID INTEGER)
RETURNS SETOF tumtip AS $$
DECLARE
    ilan tumtip;
BEGIN
    RETURN QUERY
        (SELECT n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.
        FROM notice_board n
        JOIN animal a ON n.aid = a.animal_id
        JOIN user_info u ON n.uid = u.user_id
        WHERE n.uid <> myID)
        EXCEPT
        (SELECT n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.
        FROM notice_board n
        JOIN animal a ON n.aid = a.animal_id
        JOIN user_info u ON n.uid = u.user_id
        JOIN application app ON n.notice_id = app.nid
        WHERE app.applicant_id = myID);
END;
$$ LANGUAGE 'plpgsql';
```

9. ***Sorgularınızın en az biri aggregate fonksiyonlar içermeli, having ifadesi kullanılmalıdır.***

   Func2 fonksiyonu bir aggregate fonksiyon olan count() kullanmaktadır. Bu fonksiyon aynı zamanda having ifadesi içermektedir:

```
SELECT * FROM notice_board;
CREATE OR REPLACE FUNCTION func2(town character varying)
    RETURNS bilgi[] AS$$
DECLARE
i integer;
bilgiler bilgi[];
curs cursor for SELECT COUNT(*) AS count, u.city, u.f_name, u.l_name
        FROM application app
        JOIN notice_board n ON app.nid = n.notice_id
        JOIN user_info u ON u.user_id = n.uid
        GROUP BY u.city, u.f_name, u.l_name
        HAVING town = u.city;
BEGIN
i:=0;
   FOR row_n in curs loop
    bilgiler[i]=row_n;
    i:=i+1;
    END LOOP;
    RETURN bilgiler;
END;
$$ LANGUAGE 'plpgsql'
```

10. Arayüzden girilen değerleri parametre olarak alıp ekrana sonuç döndüren 3 farklı SQL fonksiyonu tanımlamış olmalısınız. Bu fonksiyonların en az birinde "record" ve "cursor" tanımı-kullanımı olmalıdır.

SearchBySpecies, searchByCity, searchBySpeciesAndTown fonksiyonları parametre olarak arayüzden gelen değerleri almaktadır. Func2 fonksiyonu bilgi türünde değer döndüren bir fonksiyon olduğu için bilgi record'u ve curs cursor'ı tanımı-kullanımı vardır. Ayrıca tumtip, protip gibi record tanımlamaları da kullanılmaktadır.

```
CREATE OR REPLACE FUNCTION searchByTown(town VARCHAR(20), myID INTEGER)
RETURNS SETOF tumtip AS $$
DECLARE
    ilan tumtip;
BEGIN
    RETURN QUERY
        SELECT n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.ani
        FROM animal a
        JOIN notice_board n ON a.animal_id = n.aid
        JOIN user_info u ON u.user_id = n.uid
        WHERE u.city = town AND myID <> u.user_id
        EXCEPT
        SELECT n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.ani
        FROM notice_board n
        JOIN animal a ON n.aid = a.animal_id
        JOIN user_info u ON n.uid = u.user_id
        JOIN application app ON n.notice_id = app.nid
        WHERE app.applicant_id = myID
        GROUP BY n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.a
END;
$$ LANGUAGE 'plpgsql';
```

```
CREATE OR REPLACE FUNCTION searchBySpecies(tur VARCHAR(20), myID INTEGER)
RETURNS SETOF tumtip AS $$
DECLARE
    ilan tumtip;
BEGIN
    RETURN QUERY
        SELECT n.notice_id, n.aid, n.notice_date, n.sex, n.city, n.birth_date, n.breed, u.user_name, n.weight, a.species, n.anim
        FROM notice_board n
        JOIN animal a ON n.aid = a.animal_id
        JOIN user_info u ON n.uid = u.user_id
        WHERE a.species = tur AND myID <> u.user_id
        AND NOT EXISTS (
            SELECT 1
            FROM application app
            WHERE app.nid = n.notice_id AND app.applicant_id = myID
        );
END;
$$ LANGUAGE 'plpgsql';
```

```
CREATE OR REPLACE FUNCTION searchBySpeciesAndTown(tur VARCHAR(20), town VARCHAR(20), myID INTEGER)
RETURNS SETOF tumtip AS $$
DECLARE
    ilan tumtip;
BEGIN
    RETURN QUERY
        SELECT notice_id, aid, notice_date, sex, notice_board.city, notice_board.birth_date, breed, user_name,  weight, species
        FROM notice_board
        JOIN animal ON aid = animal_id
        JOIN user_info ON uid = user_id
        WHERE species = tur AND myID <> user_id

        INTERSECT

        SELECT notice_id, aid, notice_date,  sex, notice_board.city, notice_board.birth_date, breed, user_name,  weight, specie
        FROM notice_board
        JOIN animal ON aid = animal_id
        JOIN user_info ON uid = user_id
        WHERE town = notice_board.city AND myID <> user_id
        AND (notice_id, aid) NOT IN (
            SELECT n.notice_id, n.aid
            FROM notice_board n
            JOIN application app ON n.notice_id = app.nid
            WHERE app.applicant_id = myID
        );
END;
$$ LANGUAGE 'plpgsql';
-------------------------------
```

```
CREATE TYPE tumtip AS (noticeid INT, aid INT, notice_date DATE, sex CHAR, city VARCHAR(20), birth_date DATE, breed VARCHAR(20),
CREATE TYPE protip AS (product_id INT, product_name VARCHAR(20), price INT, stock INT, upload_date DATE, a_species VARCHAR(20),
CREATE TYPE mynoticetip AS (noticeid INT, uid INT, aid INT, notice_date DATE, animal_name VARCHAR(20), weight INT, sex char,cit
CREATE TYPE myapplicanttip AS (user_city varchar(20), user_fname varchar(20), user_lname varchar(20), user_name varchar(20));
CREATE TYPE nottip AS (notice_id int, uid int, aid int, notice_date date, animal_name varchar(20), weight int, sex char, city v
-------------------------------------
CREATE OR REPLACE FUNCTION func2(town character varying)
    RETURNS bilgi[] AS$$
DECLARE
i integer;
bilgiler bilgi[];
curs cursor for SELECT COUNT(*) AS count, u.city, u.f_name, u.l_name
        FROM application app
        JOIN notice_board n ON app.nid = n.notice_id
        JOIN user_info u ON u.user_id = n.uid
        GROUP BY u.city, u.f_name, u.l_name
        HAVING town = u.city;
BEGIN
i:=0;
   FOR row_n in curs loop
    bilgiler[i]=row_n;
    i:=i+1;
    END LOOP;
    RETURN bilgiler;
END;
$$ LANGUAGE 'plpgsql'
```

11. *2 adet trigger tanımlamalı ve arayüzden girilecek değerlerle tetiklemelisiniz. Trigger'ın çalıştığına dair arayüze bilgilendirme mesajı döndürülmelidir.*

trig_fonk_add_appl trigger fonksiyonu bir kullanıcı aynı ilana birden fazla kez başvurmaması için kullanılan bir triggerdır ve eğer bu durum oluşursa "Bu ilana başvurdunuz" mesajı yazdırır:

```
CREATE OR REPLACE FUNCTION trig_fonk_add_appl()
RETURNS TRIGGER AS $$
BEGIN
    IF (EXISTS (SELECT 1 FROM application WHERE NEW.nid = nid AND NEW.applicant_id = applicant_id)) THEN
        RAISE EXCEPTION 'Bu ilana başvurdunuz';
    ELSE
        RETURN NEW;
    END IF;
END;
$$ LANGUAGE 'plpgsql';
```

trig_fonk_delete_pid trigger fonksiyonu bir ürün alındığında"Ürün başarılı bir şekilde alınmıştır" mesajı yazdırır:

```
CREATE OR REPLACE FUNCTION trig_fonk_delete_pid()
RETURNS TRIGGER AS $$
BEGIN
    RAISE NOTICE'Ürün başarılı bir şekilde alınmıştır.';
    IF (OLD.stock > 1) THEN
        UPDATE products
        SET stock = stock - 1
        WHERE product_id = OLD.product_id;
        RETURN NULL; -- Returning NULL prevents the actual deletion
    END IF;
    RETURN OLD;
END;
$$ LANGUAGE 'plpgsql';
--------------------------------
CREATE TRIGGER delete_pid
BEFORE DELETE
ON products
FOR EACH ROW EXECUTE FUNCTION trig_fonk_delete_pid();
```