

# VERİ TABANI YÖNETİM SİSTEMLERİ PROJE RAPOR DOSYASI

BİLGİSAYAR VE BİLİŞİM  
FAKÜLTESİ

MUHAMMED CAN ÖZYAŞAR  
2. ÖĞRETİM C GRUBU

`muhammed.ozyasar@ogr.sa`  
`karya.edu.tr`

**ogr no: G231210009**

Proje doküman ve kodları detaylı olarak adreste yer almakta:  
<https://github.com/can-ozyasar/dataBaseProject>

## OZYASAR DERİ DATABASE PROJE AYRINTILI RAPOR DOSYASI

Ozyasar Deri & Aksesuar

*"Ozyasar Deri: Doğadan Zarafete, Kaliteden Şıklığa."*

### a) Uygulamanın Kısa Tanıtımı, İş Kuralları, İlişkisel Şema (Metinsel Gösterim)

#### 1.Uygulama Kısa Tanıtım:

Bu veri tabanı, deri ürünleri üreten İzmir’de Bornova ve Menemen ilçelerinde üretim yeri olan özyaşar deri şirketi için yazılmıştır. şirketin tüm iş süreçlerini düzenlemek amacıyla tasarlanmıştır. Şirketin personel bilgileri, müşteri ilişkileri, sipariş yönetimi ve ürün envanteri gibi veriler bu veri tabanında depolanır. Bu sayede şirket yönetimi etkili ölçüde profesyonelleşecektir.

**Personel** tablosu, çalışanların temel bilgilerini içerirken, **Calisan**, **Yoneticisi** ve **Musteritemsilcileri** tabloları daha spesifik verileri (maaş, kıdem, pozisyon, satış hedefleri) yönetir.ve temel kalıtım bu tablolar arasındadır. **Musteri** tablosu, müşterilerin bilgilerini ve hangi bölgelerde bulunduklarını takip eder. **Siparis** tablosu, müşterilerin verdiği siparişleri içerirken, **Odemetur** ve **Kargofirmasi** tabloları ise ödeme türleri ve kargo şirketleri hakkında bilgi verir.

**Deriurun** tablosu, üretilen deri ürünlerinin özelliklerini, stok miktarlarını ve fiyatlarını kaydeder. Ürünler ayrıca **Kategori**, **Tabaklamatur**, **Kaynak** ve **Yuzeyislemesi** gibi tablolarda detaylı bir şekilde sınıflandırılır.

Bu veri tabanı, şirketin operasyonel süreçlerini verimli bir şekilde yönetmek için kapsamlı bir altyapı sağlar.

#### 2.İş Kuralları:

- ❖ **Personel** sınıfı, **Çalışan** ,**Yönetici** ve **müşteri temsilcisi** sınıflarına kalıtım verir.
- ❖ Her personel yalnızca bir şehirde doğmuş olabilir, ancak bir şehirde çok sayıda personel bulunabilir.
- ❖ Personel, en az ve en çok bir **departmanda** çalışabilir ek olarak bir departmanda birden fazla personel görev alabilir.
- ❖ **Müşteri Temsilcileri** tablosundaki bir personel, birden fazla müşteri ile ilgilenebilir. Bir müşteri ise yalnızca bir müşteri temsilcisi ile ilişkilidir ve muhatap olur.
- ❖ **Bölge** bir bölgede birden fazla müşteri bulunabilir ve her müşteri yalnızca bir bölgede yer alır.

- ❖ **Sipariş** tablosunda her sipariş yalnızca bir müşteriye aittir, ancak bir müşteri birden fazla sipariş verebilir.
- ❖ **Kargo Firması**, birden fazla sipariş için kullanılabilir. Ancak, her sipariş yalnızca bir kargo firması ile gönderilir.
- ❖ Bir **Ödeme Türü**, birden fazla siparişte kullanılabilir. Her sipariş ise yalnızca bir ödeme türü ile yapılır.
- ❖ **Sipariş** tablosunda her sipariş, bir ürüne aittir, ancak bir ürün farklı siparişlerde yer alabilir.
- ❖ **Tabaklama Türü**, ürünlerin sekiz farklı kategoride sınıflandırılmasına olanak tanır. Bir ürün yalnızca bir tabaklama türüne göre üretilir ve aynı tabaklama türüne sahip birden fazla ürün olabilir.
- ❖ **Yüzey İşlemesi**, ürünlerin on farklı türde sınıflandırılmasını sağlar. Bir ürün yalnızca bir yüzey işleme türüne göre üretilir ve aynı yüzey işleme türüne sahip birden fazla ürün olabilir.
- ❖ **Kaynak Türü**, ürünlerin on farklı türde sınıflandırılmasına imkan verir. Her ürün yalnızca bir kaynak türüne göre üretilir ve aynı kaynak türüne sahip birden fazla ürün olabilir.
- ❖ **Ürünler**, birden fazla **Kategoriye** ait olabilir. Ancak her ürün yalnızca bir kategoride yer alır ve aynı kategoride birden fazla ürün olabilir.

### 3.İlişkisel Şema (Metinsel Gösterim):

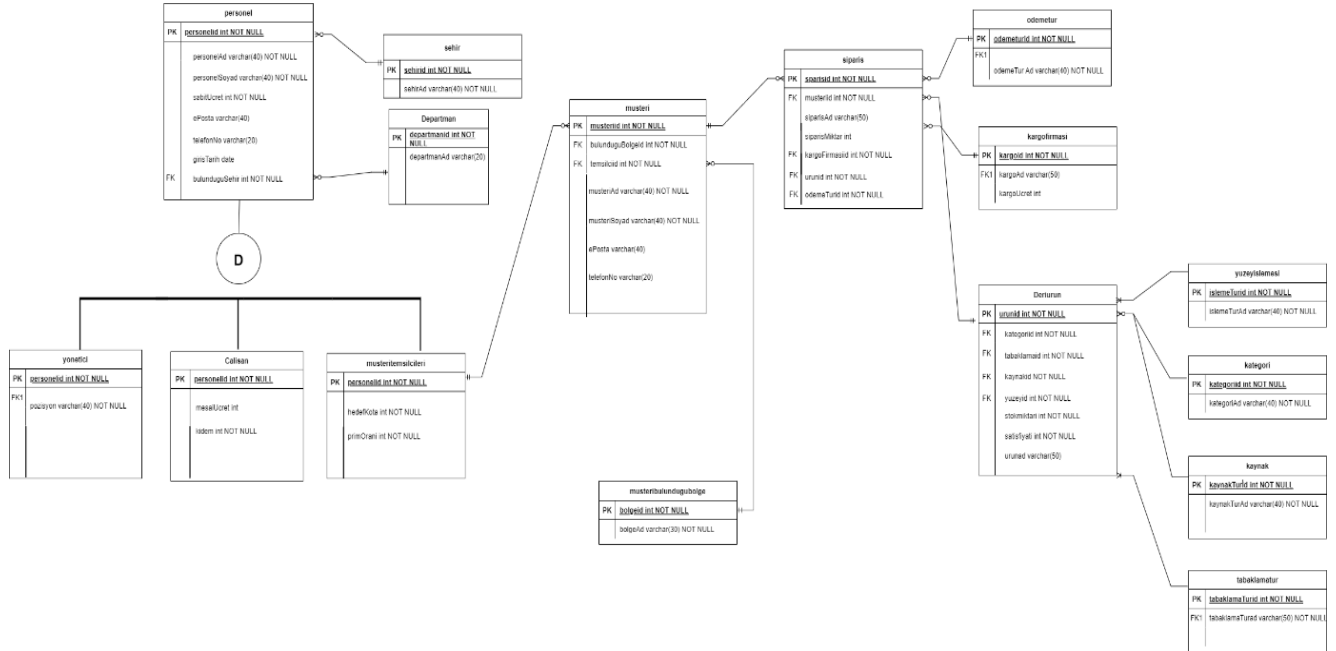
- personel (**personelid**:integer, bulundugusehir:integer, personelAd:character varying(40), personelSoyad: character varying (40), sabitUcret:integer, ePosta: character varying (40), telefonNo: character varying(20), girisTarih: date, departman varchar(20))
- calisan(**personelid**:integer, mesaiUcret:integer, kidem: integer)
- yönetici (**personelid**:integer, pozisyon: character varying(40))
- musteritemsilcileri (**personelid**:integer, hedefKota: integer , primOranı: integer)
- sehir (**sehirid**: integer, sehirAd : character varying (40))
- departman(**departmanid**:integer ,departmanAd:varchar(20))
- müşteri(**musteriid**:integer, bulundugubolge\_id:integer , temsilciid : integer , müşteriAd: String , müşteriSoyad: String, telefonNo:numeric , ePostaAdresi: character varying (20))
- musteribulundugubolge ( **bolgeid**: integer, bolgeAd:character varying (30))
- siparis (**siparisid**: integer, urunid :integer , kargoFirmasiid :integer ,musteriid:integer, ödemeTurid: integer, siparisAd: character varying (50) , siparisMiktar: integer )
- odemetur (**odemeturid**: integer, odemeturAd: character varying(40))
- kargofirmasi (**kargoid**: integer, kargoAd: character varying (30) , kargoUcret: integer)
- deriurun (**urunid**: integer, kategoriid:integer, tabaklamaid:integer, kaynakid: integer,
- yuzeyid:integer , stokmiktar:integer, satisfiyati: integer, urunad: character varying(50))
- kategori (**kategoriid** : integer , kategoriAd: character varying(50))

- tabaklamatur (**tabaklamaTurid**: integer , tabaklamaTurAd: character varying (50))
- kaynak (**kaynakTurid** : integer , kaynakTurAd: character varying (50))
- yuzeyislemesi (**islemeTurid** : integer , islemeTurAd)

## b) Varlık Bağntı Modeli (Crow's Foot, Kalıtım)

**Kalıtım:** Yöneticiler, çalışanlar ve müşteri temsilcileri; Personel tablosundan kalıtım alınır. Yani her alt tablo olan yönetici, çalışan ve müşteri temsilcisi, personel tablosunda yer alan bilgileri taşıırken, aynı zamanda kendine has bilgilere de sahip olur.

- Discont kalıtım
  - Gösterim: D
  - Discont kalıtım kullandım Aynı temel varlıkla ilgili tek bir çocuk varlık olabilir yani bir personel aynı anda hem çalışan hem yönetici olamaz
- Kısmi Bütünlük
  - Gösterim: Tek Çizgi
  - Üst tip kayıtlar, alt tip kayıtlar olmadan da mevcut olabilir.



\*\*Ekstra olarak Dosya içinde png olarak resmi mevcut

## c) Veri tabanını, İçerisindeki Verilerle Birlikte Oluşturmayı Sağlayan SQL İfadeleri

## PERSONEL TABLOSU

```
CREATE TABLE IF NOT EXISTS public.personel
(
    personelid integer NOT NULL,
    "personelAd" character varying(40) NOT NULL,
    "personelSoyad" character varying(40) NOT NULL,
    "sabitUcret" integer NOT NULL,
    "ePosta" character varying(40),
    "telefonNo" character varying(20),
    "girisTarih" date,
    "bulunduguSehir" integer NOT NULL,
    departman integer,
    CONSTRAINT personel_pkey PRIMARY KEY (personelid),
    CONSTRAINT departman_foreign FOREIGN KEY (departman)
        REFERENCES public.departman (departmanid),
    CONSTRAINT sehir_foreign FOREIGN KEY ("bulunduguSehir")
        REFERENCES public.sehir (sehirid)
);
```

```
CREATE INDEX IF NOT EXISTS fki_sehir_foreign
    ON public.personel USING btree ("bulunduguSehir");
```

## SEHİR TABLOSU

```
CREATE TABLE IF NOT EXISTS public.sehir
(
    sehirid integer NOT NULL,
    "sehirAd" character varying(40) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT sehir_pkey PRIMARY KEY (sehirid)
)
```

## VERİ EKLEME :

```
INSERT INTO public.sehir (sehirid, sehirAd)
```



```
VALUES (1, 'Adana'), (2, 'Adıyaman'), (3, 'Afyonkarahisar'), (4, 'Ağrı') (5, 'Amasya'), (6, 'Ankara'), (7, 'Antalya'), (8, 'Artvin'), (9, 'Aydın'), (10, 'Balıkesir'), (11, 'Bilecik'), (12, 'Bingöl'), (13, 'Bitlis'), (14, 'Bolu'), (15, 'Burdur'), (16, 'Bursa'), (17, 'Çanakkale'), (18, 'Çankırı'), (19, 'Çorum'), (20, 'Denizli'), (21, 'Diyarbakır'), (22, 'Edirne'), (23, 'Elazığ'), (24, 'Erzincan'), (25, 'Erzurum'), (26, 'Eskişehir'),(27, 'Gaziantep'), (28, 'Giresun'), (29, 'Gümüşhane'), (30, 'Hakkari');
```

## DEPARTMAN TABLOSU

```
CREATE TABLE IF NOT EXISTS public.departman
```

```
(  
    departmanid integer PRIMARY KEY,  
    departmanad character varying(20) NOT NULL  
);
```

## DEPARTMAN EKLEME

```
INSERT INTO "public"."departman" ("departmanid","departmanad")
```

```
VALUES
```

```
(8,'Hukuk '), (7,'Halkla ilişkiler '), (6,'Ar-Ge'), (5,'İnsan kaynakları'), (4,'Muhasebe '), (3,'Finans'), (2,'Pazarlama'), (1,'ÜRETİM');
```

## YÖNETİCİ TABLOSU

```
CREATE TABLE IF NOT EXISTS public.yonetici
```

```
(  
    pozisyon character varying(40) NOT NULL,  
    CONSTRAINT yonetici_pkey PRIMARY KEY (personelid)  
)  
INHERITS (public.personel);
```

## YÖNETİCİ EKLEME

```
INSERT INTO public.yonetici (personelid, "personelAd", "personelSoyad", "sabitUcret", "ePosta", "telefonNo", "girisTarih", "bulunduguSehir", "pozisyon", departman)
```

```
VALUES
```

```
(1, 'Ali', 'Yılmaz', 5000, 'ali.yilmaz@example.com', '05551234567', '2024-01-01', 34, 'Müdür', 1),  
(2, 'Ayşe', 'Kara', 6000, 'ayse.kara@example.com', '05552345678', '2023-12-15', 35, 'Müdür Yardımcısı', 2),  
(3, 'Mehmet', 'Demir', 5500, 'mehmet.demir@example.com', '05553456789', '2023-11-10', 06, 'Uzman', 3),  
(4, 'Fatma', 'Çelik', 5800, 'fatma.celik@example.com', '05554567890', '2023-10-20', 01, 'Uzman Yardımcısı', 4),
```

\*\*\* toplam 20 tane ekledim burada yer kaplamaması açısından yazmadım

## CALISAN TABLOSU

```
CREATE TABLE IF NOT EXISTS public.calisan
(
    "mesaiUcret" integer,
    kidem integer NOT NULL,
    CONSTRAINT calisan_pkey PRIMARY KEY (personelid)
)
INHERITS (public.personel);
```

```
CREATE OR REPLACE TRIGGER tr_maas_guncelle
AFTER UPDATE OF "mesaiUcret"
ON public.calisan
FOR EACH ROW
EXECUTE FUNCTION public.maas_guncelle();
```

## ÇALIŞAN EKLEME

```
INSERT INTO public.calisan (personelid, "personelAd", "personelSoyad", "sabitUcret", "ePosta", "telefonNo", "girisTarih",
"bulunduguSehir", "mesaiUcret", kidem, departman)
```

VALUES

```
(21, 'Ece', 'Aydın', 5200, 'ece.aydin@example.com', '05551112233', '2024-03-01', 34, 100, 3, 2),
(22, 'Kerem', 'Çetin', 5600, 'kerem.cetin@example.com', '05552233444', '2024-02-20', 06, 120, 2, 3),
(23, 'Sude', 'Yüce', 5800, 'sude.yuce@example.com', '05553344555', '2024-01-15', 01, 110, 4, 1),
(24, 'Caner', 'Bozkurt', 6000, 'caner.bozkurt@example.com', '05554455666', '2023-12-10', 16, 130, 5, 4);
```

*//Toplam 15 tane ekledim yer kapladığı için buraya yazmadım*

## MUSTERİ TEMSİLCİLERİ TABLOSU

```
CREATE TABLE IF NOT EXISTS public.musteritemsilcileri
(
    "hedefKota" integer NOT NULL,
    "primOranı" integer NOT NULL,
    CONSTRAINT "musteriTemsicileri_pkey" PRIMARY KEY (personelid)
)
INHERITS (public.personel);
```

## TEMSİLCİ EKLEME

```
INSERT INTO public.musteritemsilcileri (personelid, "personelAd", "personelSoyad", "sabitUcret", "ePosta", "telefonNo", "girisTarih", "bulunduguSehir", "hedefKota", "primOranı", departman)
```

VALUES

```
(36, 'Elif', 'Çakır', 5400, 'elif.cakir@example.com', '05551122333', '2024-05-15', 34, 10000, 8, 1),
```

```
(37, 'Murat', 'Güzel', 5600, 'murat.guzel@example.com', '05552233444', '2024-04-10', 06, 15000, 10, 2),
```

```
(38, 'Ali', 'Demir', 5700, 'ali.demir@example.com', '05553344555', '2024-03-05', 01, 12000, 7, 3),
```

```
(39, 'Ayşe', 'Kılıç', 5900, 'ayse.kilic@example.com', '05554455666', '2024-02-25', 16, 11000, 9, 4),
```

*//Toplam 15 tane ekledim yer kapladığı için buraya yazmadım*

## MÜŞTERİ TABLOSU

```
CREATE TABLE IF NOT EXISTS public.musteri
```

(

```
    musteriid integer NOT NULL,
```

```
    "musteriAd" character varying(40) NOT NULL,
```

```
    "musteriSoyad" character varying(40) NOT NULL,
```

```
    temsilciid integer NOT NULL,
```

```
    "bulunduguBolgeid" integer NOT NULL,
```

```
    "telefonNo" character varying(20),
```

```
    "ePostaAdresi" character varying(30),
```

```
    CONSTRAINT muster_i_pkey PRIMARY KEY (musteriid),
```

```
    CONSTRAINT bolge_foreign FOREIGN KEY ("bulunduguBolgeid")
```

```
        REFERENCES public.musteribulundugubolge (bolgeid),
```

```
    CONSTRAINT temsilci_foreign FOREIGN KEY (temsilciid)
```

```
        REFERENCES public.musteritemsilcileri (personelid)
```

);

```
CREATE INDEX IF NOT EXISTS fki_bolge_foreign
```

```
    ON public.musteri USING btree ("bulunduguBolgeid");
```

```
CREATE INDEX IF NOT EXISTS fki_temsilci_foreign
```

```
    ON public.musteri USING btree (temsilciid);
```

```
CREATE OR REPLACE TRIGGER telefon_no_trigger
```

```
    BEFORE INSERT ON public.musteri
```



FOR EACH ROW

EXECUTE FUNCTION public.telefon\_no\_kontrol();

CREATE OR REPLACE TRIGGER temsilci\_kontrol\_trigger

BEFORE INSERT ON public.musteri

FOR EACH ROW

EXECUTE FUNCTION public.temsilci\_kontrol();

## MÜŞTERİ EKLEME

INSERT INTO public.musteri (musteriid, "musteriAd", "musteriSoyad", temsilciid, "bulunduguBolgeid", "telefonNo", "ePostaAdresi")

VALUES

(1, 'Ahmet', 'Yılmaz', 1, 1, '0543 123 45 67', 'ahmet.yilmaz@example.com'),

(2, 'Mehmet', 'Kaya', 2, 2, '0543 234 56 78', 'mehmet.kaya@example.com'),

(3, 'Ayşe', 'Çelik', 3, 3, '0543 345 67 89', 'ayse.celik@example.com'),

(4, 'Fatma', 'Demir', 4, 4, '0543 456 78 90', 'fatma.demir@example.com'),

(5, 'Ali', 'Kara', 5, 5, '0543 567 89 01', 'ali.kara@example.com'),

*//Toplam 20 tane ekledim yer kapladığı için buraya yazmadım*

## KATEGORİ TABLOSU

CREATE TABLE IF NOT EXISTS public.kategori

(

kategoriid integer NOT NULL,

"kategoriAd" character varying(50) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT kategori\_pkey PRIMARY KEY (kategoriid)

)

## KATEGORİ EKLEME

INSERT INTO public.kategori (kategoriid, "kategoriAd")

VALUES(1, 'El İşi'),(2, 'Giyim'),(3, 'Döşeme'),(4, 'Spor Ekipmanları'),(5, 'Sanayi Ürünleri'),(6, 'Yapı Malzemeleri'),(7, 'Elektronik'),(8, 'Gıda'),(9, 'Mobilya'),(10, 'Otomotiv'),(11, 'Teknolojik Ürünler'),(12, 'Züccaciye'),(13, 'Hobi Ürünleri'),(14, 'Sağlık Ürünleri'),(15, 'Ev Gereçleri');

## MÜŞTERİ BULUNDUĞU BÖLGE TABLOSU

```
CREATE TABLE IF NOT EXISTS public.musteribulundugubolge
(
    bolgeid integer NOT NULL,
    "bolgeAd" character varying(30) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT musteribulundugubolge_pkey PRIMARY KEY (bolgeid)
)
```

### BÖLGE EKLME

```
INSERT INTO public.musteribulundugubolge ("bolgeid","bolgeAd") VALUES(1,'afrika') , (2,'Avrupa'), (3,'Asya'),
(4,'Avusturalya')
```

## KAYNAK TABLOSU

```
CREATE TABLE IF NOT EXISTS public.kaynak
(
    "kaynakTurid" integer NOT NULL,
    "kaynakTurAd" character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT kaynak_pkey PRIMARY KEY ("kaynakTurid")
)
```

### KAYNAK EKLEME

```
INSERT INTO public.kaynak ("kaynakTurid", "kaynakTurAd")
VALUES(1, 'Dana Derisi'),(2, 'Yılan Derisi'),(3, 'Sentetik Deri'),(4, 'Kuzu Derisi'),(5, 'Tavşan Derisi'),(6, 'Yumuşak Deri'),(7,
'Koyun Derisi'),(8, 'Timsah Derisi'),(9, 'Vegan Deri'),(10, 'Sığır Derisi');
```

## KARGO FİRMASI TABLOSU

```
CREATE TABLE IF NOT EXISTS public.kargofirmasi
(
    kargoid integer NOT NULL,
    "kargoAd" character varying(30) COLLATE pg_catalog."default" NOT NULL,
    "kargoUcret" integer NOT NULL,
    CONSTRAINT kargofirmasi_pkey PRIMARY KEY (kargoid)
)
```

## KARGO FİRMASI EKLEME

```
INSERT INTO public.kargofirmani (kargoid, kargoAd, kargoUcret)
```

```
VALUES
```

```
(1, 'Aras Kargo', 15),(2, 'Yurtiçi Kargo', 18),(3, 'MNG Kargo', 20),(4, 'PTT Kargo', 10),(5, 'Sürat Kargo', 12),(6, 'UPS', 25),(7, 'DHL', 30),(8, 'FedEx', 35),(9, 'KargoNet', 22),(10, 'TNT', 28);
```

## SİPARİŞ TABLOSU

```
CREATE TABLE IF NOT EXISTS public.siparis
```

```
(  
    siparisid integer NOT NULL,  
    musteriid integer NOT NULL,  
    "siparisAd" character varying(50),  
    "siparisMiktar" integer,  
    "kargoFirmasiid" integer NOT NULL,  
    urunid integer NOT NULL,  
    "odemeTurid" integer NOT NULL,  
    CONSTRAINT siparis_pkey PRIMARY KEY (siparisid),  
    CONSTRAINT kargo_foreign FOREIGN KEY ("kargoFirmasiid")  
        REFERENCES public.kargofirmani (kargoid),  
    CONSTRAINT musteri_foreign FOREIGN KEY (musteriid)  
        REFERENCES public.musteri (musteriid),  
    CONSTRAINT odeme_foreign FOREIGN KEY ("odemeTurid")  
        REFERENCES public.odemetur (odemeturid),  
    CONSTRAINT urun_foreign FOREIGN KEY (urunid)  
        REFERENCES public.deriyurun (urunid)  
);
```

```
CREATE INDEX IF NOT EXISTS fki_kargo_foreign  
    ON public.siparis USING btree ("kargoFirmasiid");
```

```
CREATE INDEX IF NOT EXISTS fki_musteri_foreign  
    ON public.siparis USING btree (musteriid);
```

```
CREATE INDEX IF NOT EXISTS fki_odeme_foreign  
    ON public.siparis USING btree ("odemeTurid");
```

```
CREATE INDEX IF NOT EXISTS fki_urun_foreign
ON public.siparis USING btree (urunid);
```

```
CREATE OR REPLACE TRIGGER odeme_turu_kontrol_trigger
BEFORE INSERT OR UPDATE
ON public.siparis
FOR EACH ROW
EXECUTE FUNCTION public.odeme_turu_kontrol();
```

```
CREATE OR REPLACE TRIGGER siparis_stok_guncelle_trigger
AFTER INSERT
ON public.siparis
FOR EACH ROW
EXECUTE FUNCTION public.stok_guncelle();
```

```
CREATE OR REPLACE TRIGGER stok_kontrol_trigger
BEFORE INSERT
ON public.siparis
FOR EACH ROW
EXECUTE FUNCTION public.stok_kontrol();
```

## SİPARİŞ EKLEME

```
INSERT INTO public.siparis (siparisid, musteriid, "siparisAd", "siparisMiktar", "kargoFirmasiid", urunid, "odemeTurid")
VALUES(1, 1, 'Deri Ceket', 1, 1, 1, 1),(2, 2, 'Deriden Yapılmış Çanta', 2, 2, 2, 2),(3, 3, 'Yılan Derisi Çanta', 1, 3, 3, 3),(4, 4,
'Süet Ceket', 2, 4, 4, 4),(5, 5, 'Deriden Ayakkabı', 1, 1, 5, 1)
```

## TABAKLAMA TURU TABLOSU

```
CREATE TABLE IF NOT EXISTS public.tabaklamatur
(
    "tabaklamaTurid" integer NOT NULL,
    "tabaklamaTurAd" character varying(50) COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT tabaklamatur_pkey PRIMARY KEY ("tabaklamaTurid")
)
```

## TABAKLAMA TÜR EKLEME

```
INSERT INTO public.tabaklamatur ("tabaklamaTurid", "tabaklamaTurAd")
```

```
VALUES(1, 'Derin Tabaklama'),(2, 'Yüzey Tabaklama'),(3, 'Boya Tabaklaması'),(4, 'Sünger Tabaklama'),(5, 'Yağlı Tabaklama'),(6, 'Su Bazlı Tabaklama'),(7, 'Kimyasal Tabaklama'),(8, 'Isıl İşlem Tabaklama');
```

## YÜZEY İŞLEME TABLOSU

```
CREATE TABLE IF NOT EXISTS public.yuzeyislemesi
```

```
(  
    "islemeTurid" integer NOT NULL,  
    "islemeTurAd" character varying(40) COLLATE pg_catalog."default" NOT NULL,  
    CONSTRAINT yuzeyislemesi_pkey PRIMARY KEY ("islemeTurid")  
)
```

## YÜZEY EKLEME

```
INSERT INTO public.yuzeyislemesi ("islemeTurid", "islemeTurAd")
```

```
VALUES
```

```
(1, 'Pürüzsüz Deri'),(2, 'Süet Deri'),(3, 'Doğal Deri'),(4, 'Boya Derisi'),(5, 'Parlatılmış Deri'),(6, 'Yanmış Deri'),  
(7, 'Kadife Deri'),(8, 'Sarımsak Derisi'),(9, 'Krem Derisi'),(10, 'Mat Deri');
```

## ODEME TUR TABLOSU

```
CREATE TABLE IF NOT EXISTS public.odemetur
```

```
(  
    odemeturid integer NOT NULL,  
    "odemeturAd" character varying(40) COLLATE pg_catalog."default" NOT NULL,  
    CONSTRAINT odemetur_pkey PRIMARY KEY (odemeturid)  
)
```

## ODEME EKLEME

```
INSERT INTO public.odemetur ("odemeturid", "odemeturAd")
```

```
VALUES
```

```
(1, 'Nakit'),(2, 'Kredi Kartı'),(3, 'Banka Havalesi'),(4, 'Kapıda Ödeme'),(5, 'Paypal'),(6, 'EFT'),(7, 'Kredi Kartı ile Taksitli'),(8, 'Mobil Ödeme'),(9, 'Bitcoin'),(10, 'Alışveriş Kredisi');
```

## DERİ URUN TABLOSU

```
CREATE TABLE IF NOT EXISTS public.derirun(  
    urunid integer NOT NULL,  
    kategoriid integer NOT NULL,  
    tabaklamaid integer NOT NULL,  
    kaynakid integer NOT NULL,  
    yuzeyid integer NOT NULL,  
    stokmiktari integer NOT NULL,  
    satisfiyati integer NOT NULL,  
    urunad character varying(50),  
    CONSTRAINT derirun_pkey PRIMARY KEY (urunid),  
    CONSTRAINT kategori_foreign FOREIGN KEY (kategoriid)  
        REFERENCES public.kategori (kategoriid),  
    CONSTRAINT kaynak_foreign FOREIGN KEY (kaynakid)  
        REFERENCES public.kaynak ("kaynakTurid"),  
    CONSTRAINT tabaklama_foreign FOREIGN KEY (tabaklamaid)  
        REFERENCES public.tabaklamatur ("tabaklamaTurid"),  
    CONSTRAINT yuzeyisleme_foreign FOREIGN KEY (yuzeyid)  
        REFERENCES public.yuzeyislemesi ("islemeTurid")  
);
```

```
CREATE INDEX IF NOT EXISTS fki_kategori_foreign  
    ON public.derirun USING btree (kategoriid);
```

```
CREATE INDEX IF NOT EXISTS fki_kaynak_foreign  
    ON public.derirun USING btree (kaynakid);
```

```
CREATE INDEX IF NOT EXISTS fki_tabaklama_foreign  
    ON public.derirun USING btree (tabaklamaid);
```

```
CREATE INDEX IF NOT EXISTS fki_yuzeyisleme_foreign  
    ON public.derirun USING btree (yuzeyid);
```

## **DERİ ÜRÜN EKLEME**



```
INSERT INTO public.deriurun ("urunid", "kategoriid", "tabaklamaid", "kaynakid", "yuzeyid", "stokmiktar", "satisfiyati", "urunad")
```

```
VALUES
```

```
(1, 1, 1, 1, 1, 100, 500, 'Klasik Deri Ceket'),
```

```
(2, 1, 2, 2, 2, 150, 600, 'Süet Ceket'),
```

```
(3, 2, 3, 3, 1, 200, 550, 'Deri Koltuk'),
```

```
(4, 3, 1, 4, 3, 80, 700, 'Deri Çanta'),
```

```
(5, 4, 2, 5, 2, 120, 450, 'Deri Ayakkabı'),
```

*//Toplam 20 tane ekledim yer kapladığı için buraya yazmadım*

## d)FONKSİYON VE TRİGGERLER

### 1-siparis tablosunu uygun isimlerle doldurup döndüren fonksiyon

**NE YAPIYOR :**Sipariş tablosu yapı gereği id leri tutuyor bu durumu personellerin daha anlayabilmesi için uygun isimleri uygun sütunlara yazıp tabloyu döndürüyor .

```
CREATE OR REPLACE FUNCTION siparis_bilgilerini_goster()
```

```
RETURNS TABLE(
```

```
    siparisid INTEGER,
```

```
    musteriAd VARCHAR,
```

```
    musteriSoyad VARCHAR,
```

```
    siparisAd VARCHAR,
```

```
    siparisMiktar INTEGER,
```

```
    kargoFirmaAd VARCHAR,
```

```
    urunAd VARCHAR,
```

```
    ödemeTurAd VARCHAR
```

```
) AS $$
```

```
BEGIN
```

```
    RETURN QUERY
```

```

SELECT
    s.siparisid,
    m."musteriAd",
    m."musteriSoyad",
    s."siparisAd",
    s."siparisMiktar",
    k."kargoAd", -- Kargo Firması Adı
    u."urunad", -- Ürün Adı
    o."odemeturAd" -- Ödeme Türü Adı
FROM
    public.siparis s
JOIN
    public.musteri m ON s.musteriid = m.musteriid
JOIN
    public.kargofirmasi k ON s."kargoFirmasiid" = k.kargoid
JOIN
    public.deriuurun u ON s.urunid = u.urunid
JOIN
    public.odemetur o ON s."odemeTurid" = o.odemeturid;
END;
$$ LANGUAGE plpgsql;

```

## KULLANIMI

```
SELECT * FROM siparis_bilgilerini_goster();
```

## 2-PERSONELLERİN ŞİRKETE OLAN MALİYETİNİ HESAPLAYAN FONKSİYON VE KULLANIMI

**NE YAPIYOR :** personel tablosunda bulunan tüm personellerin şirkete olan toplam maliyetini hesaplıyor 1 personel için ortalama maliyet maaş x 2,3 olarak hesaplanır( vergi sgk vs )

```

CREATE OR REPLACE FUNCTION toplam_maas_hesapla()
RETURNS NUMERIC AS $$
DECLARE
    toplam_maas NUMERIC;
BEGIN
    SELECT SUM("sabitUcret" * 2.3) INTO toplam_maas FROM personel;

```

```
    RETURN toplam_maas;  
END;  
$$ LANGUAGE plpgsql;
```

#### KULLANIMI:

```
SELECT toplam_maas_hesapla();
```

### 3-KOTA YÜZDESİ HESAPLAYAN FONKSİYON

**NE YAPIYOR:** Müşteri temsilcisinin güncel performansını ölçüyor hedef kotanın ne oranda tamamlandığını döndürüyor % olarak

```
CREATE OR REPLACE FUNCTION public.hedef_kota_yuzdesi(  
    p_personelid integer,  
    p_gerceklesen integer)  
RETURNS numeric  
LANGUAGE 'plpgsql'  
AS $BODY$  
DECLARE  
    hedef_Kota INTEGER;  
    yuzde DECIMAL;  
BEGIN  
    SELECT "hedefKota" INTO hedef_Kota  
    FROM musteritemsilcileri  
    WHERE personelid = p_personelid;  
  
    -- NULL veya sıfır kontrolü  
    IF hedef_Kota IS NULL OR hedef_Kota = 0 THEN  
        RETURN 0;  
    END IF;  
  
    -- Yüzde hesaplama  
    yuzde := (p_gerceklesen * 100.0) / hedef_Kota;
```

```
RETURN yuzde;  
END;  
$BODY$;
```

## KULLANNIMI

```
SELECT hedef_kota_yuzdesi(36, 500);  
  
//36 id li temsilci 500 müşteri ile ilgilenmiş buna göre % kaç tamamladı kotasını
```

## 4 ÜRÜN ZAM YAP

**NE YAPIYOR:** Verilen id deki ürün e verilen % oranında zam yapıp fiyatı güncelliyor.

```
CREATE OR REPLACE FUNCTION public.urun_Zam_Yap(  
    p_urunid INTEGER,  
    p_yuzde NUMERIC  
) RETURNS VOID  
LANGUAGE plpgsql  
AS $BODY$  
BEGIN  
    -- Yüzde kontrolü  
    IF p_yuzde <= 0 THEN  
        RAISE EXCEPTION 'Yüzde değeri pozitif olmalıdır!';  
    END IF;  
  
    -- Ürün fiyatını artır  
    UPDATE deriurun  
    SET satisfiyati = satisfiyati + (satisfiyati * p_yuzde / 100)  
    WHERE urunid = p_urunid;  
  
    -- Eğer ürün bulunamazsa  
    IF NOT FOUND THEN  
        RAISE NOTICE 'Belirtilen ID % için ürün bulunamadı!', p_urunid;  
    ELSE
```

```
        RAISE NOTICE 'Ürün ID % için fiyat % oranında artırıldı.', p_urunid, p_yuzde;

    END IF;

END;

$BODY$;
```

## KULLANIMI

```
SELECT urun_Zam_Yap(1, 20); -- Ürün ID 1'in fiyatını %20 artır.
```

## 5 SİPARİS TOPLAM DEĞER

**NE YAPIYOR:** Parametre olarak verilen ürün id'yi kullanarak siparişteki ürünün tablosundan fiyatını bulup adet ile çarparak toplam değerini hesaplıyor ,yani işletmeye ödenecek para miktarı.

```
CREATE OR REPLACE FUNCTION public.siparis_toplam_degeri( p_siparisid integer )

    RETURNS numeric

    LANGUAGE 'plpgsql'

    COST 100

    VOLATILE PARALLEL UNSAFE

AS $BODY$

DECLARE

    toplam_deger NUMERIC;

BEGIN

    SELECT SUM(s."siparisMiktar" * u.satisfiyati)

    INTO toplam_deger

    FROM siparis s

    JOIN deriurun u ON s.urunid = u.urunid

    WHERE s.siparisid = p_siparisid;

    IF toplam_deger IS NULL THEN

        RETURN 0;

    END IF;

    RETURN toplam_deger;

END;

$BODY$;
```

## KULLANIMI

siparis\_toplam\_degeri(1) // 1 İD Lİ SİPARİS DEĞERİ URUN FİYATI KULLANRAK ADET İLE ÇARPAR

## 6 MAAŞ YÜZDE ARTTIR

**NE YAPIYOR:** Belirtilen müşteri id li müşterinin maaşına belirtilen oranda zam yapıyor maaşı güncelliyor

```
CREATE OR REPLACE FUNCTION public.maas_yuzde_artir( personel_id integer , yuzde numeric )
```

```
RETURNS void
```

```
LANGUAGE 'plpgsql'
```

```
COST 100
```

```
VOLATILE PARALLEL UNSAFE
```

```
AS $BODY$
```

```
BEGIN
```

```
IF yuzde <= 0 THEN
```

```
    RAISE EXCEPTION 'Yüzde değeri pozitif olmalıdır!';
```

```
END IF;
```

```
UPDATE personel
```

```
SET "sabitUcret" = "sabitUcret" + ("sabitUcret" * yuzde / 100)
```

```
WHERE "personelid" = personel_id;
```

```
IF NOT FOUND THEN
```

```
    RAISE NOTICE 'Belirtilen ID"ye sahip çalışan bulunamadı!';
```

```
END IF;
```

```
END;
```

```
$BODY$;
```

## KULLANIMI

maas\_yuzde\_artir(2, 20) // 2 İD Lİ PERSONELE % 20 ZAM YAPAR

## TRİGGERLER

### 1-MESAİ ÜCRETİ DEĞİŞTİĞİ ZAMAN MAAŞI GÜNCELLE



**NE YAPIYOR:** çalışan tablosunda mesai ücreti değiştiği zaman sabit alacağı maaş ücretini de arttırıyor 10 ile çarparak

```
CREATE OR REPLACE FUNCTION public.maas_guncelle()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    -- Mesai ücreti değiştiyse, sabit ücreti güncelle
    IF NEW."mesaiUcret" <> OLD."mesaiUcret" THEN
        UPDATE public.personel
        SET "sabitUcret" = "sabitUcret" + (NEW."mesaiUcret" * 10)
        WHERE personelid = NEW.personelid;
    END IF
    RETURN NEW;
END;
$BODY$;
```

## TRIGGER

```
CREATE OR REPLACE TRIGGER maas_kontrol_trigger
    BEFORE INSERT OR UPDATE
    ON public.calisan
    FOR EACH ROW
    EXECUTE FUNCTION public.maas_kontrol();
```

## 2-SİPARİŞ OLUŞTRULDUKTAN SONRA STOK AZALTMA (GÜNCELLEME)

**NE YAPIYOR:** sipariş oluşturma işlemi eğer başarılı olursa deriurun tablosundan ilgili sipariş değerini stoktan azaltarak stok miktarının güncel kalmasını sağlıyor

```
CREATE OR REPLACE FUNCTION public.stok_guncelle()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
```

```

VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

    IF (SELECT stokmiktari FROM public.deriuurun WHERE urunid = NEW.urunid) < NEW."siparisMiktar" THEN

        RAISE EXCEPTION 'Yeterli stok yok!';

    ELSE

        -- Sipariş edilen ürünün stok miktarını azalt

        UPDATE public.deriuurun

        SET stokmiktari = stokmiktari - NEW."siparisMiktar"

        WHERE urunid = NEW.urunid;

    END IF;

    RETURN NEW;

END;

$BODY$;

```

## TRIGGER

```

CREATE OR REPLACE TRIGGER siparis_stok_guncelle_trigger

AFTER INSERT

ON public.siparis

FOR EACH ROW

EXECUTE FUNCTION public.stok_guncelle();

```

## 3-SİPARİŞE EKLENMEDEN ÖNCE STOK KONTROLÜ

**NE YAPIYOR:** sipariş ekleme işleminden önce yeterli stok varmı diye kontrol ediyor eğer stok yetersizse siparişin oluşmasına izin vermiyor

```

CREATE OR REPLACE FUNCTION public.stok_kontrol()

RETURNS trigger

LANGUAGE 'plpgsql'

COST 100

VOLATILE NOT LEAKPROOF

AS $BODY$

```

```

BEGIN

    IF NEW."siparisMiktar" > (SELECT stokmiktar FROM public.deriurun WHERE urunid = NEW.urunid) THEN

        RAISE EXCEPTION 'Stok miktarı yetersiz!';

    END IF;

    RETURN NEW;

END;

$BODY$;

```

## TRİGGER

```

CREATE OR REPLACE TRIGGER stok_kontrol_trigger

    BEFORE INSERT OR UPDATE

    ON public.siparis

    FOR EACH ROW

    EXECUTE FUNCTION public.stok_kontrol();

```

## 4-MAAŞ VERMEDE ALT SINIR KONTROLÜ

**NE YAPIYOR:** maaş belirleme işleminde ekleme yada güncelleme işleminde maaş alt sınırı 1000 tl olacak şekilde daha altında maaş girilmesini önüyor

```

CREATE OR REPLACE FUNCTION public.maas_kontrol()

    RETURNS trigger

    LANGUAGE 'plpgsql'

    COST 100

    VOLATILE NOT LEAKPROOF

AS $BODY$

BEGIN

    -- Maaşın 1000 TL'nin altında olup olmadığını kontrol et

    IF NEW."sabitUcret" < 1000 THEN

        RAISE EXCEPTION 'Maaş 1000 TL'nin altında olamaz!';

    END IF;

    RETURN NEW;

```

```
END;  
$BODY$;
```

## TRIGGER

```
CREATE OR REPLACE TRIGGER maas_kontrol_trigger  
    BEFORE INSERT OR UPDATE  
    ON public.personel  
    FOR EACH ROW  
    EXECUTE FUNCTION public.maas_kontrol()
```

## 5-SİPARİŞ SİLME SONRASI STOK ARTTIRMA

**NE YAPIYOR:** eğer sipariş silinirse yada iptal edilirse siparişte bulunan ürünleri tekrar stok miktarına ekliyor stok artırıyor.

```
CREATE OR REPLACE FUNCTION public.stok_artir()  
    RETURNS trigger  
    LANGUAGE 'plpgsql'  
    COST 100  
    VOLATILE NOT LEAKPROOF  
AS $BODY$  
BEGIN  
    -- Sipariş silindiğinde, sipariş miktarını stok miktarına ekle  
    UPDATE public.deriyurun  
    SET stokmiktar = stokmiktar + OLD."siparisMiktar"  
    WHERE urunid = OLD.urunid;  
  
    RETURN OLD;  
END;  
$BODY$;
```

## TRIGGER

```
CREATE OR REPLACE TRIGGER stok_artir_trigger  
    AFTER DELETE ON public.siparis
```

FOR EACH ROW

EXECUTE FUNCTION public.stok\_artir();

## e)Arama, Ekleme, Silme, Güncelleme işlemlerine ait ekran görüntüleri

### 1-ARAMA İŞLEMİ:

personelid	personelAd	personelSoyad	sabitUcret	ePosta	telefonNo	girisTarih
22	Kerem	Çetin	5600	kerem.cetin@exa...	5552233444	20.02.2024
23	Sude	Yüce	5800	sude.yuce@exa...	5553344555	15.01.2024
24	Caner	Bozkurt	6000	caner.bozkurt@e...	5554455666	10.12.2023
25	Melis	Koc	6100	melis.koc@exam...	5555566777	5.11.2023
26	Burak	Efe	4900	burak.efe@exam...	5556677888	1.10.2023
27	Selin	Aslan	5300	selin.aslan@exa...	5557788999	20.09.2023
28	Onur	Yilmaz	5800	onur.yilmaz@exa...	5558899000	15.08.2023
29	Deniz	Gül	6200	deniz.gul@exam...	5559900111	1.07.2023
30	Zeynep	Şentürk	5900	zeynep.senturk@...	5550011222	15.06.2023
31	Batuhan	Kaya	5700	batuhan.kaya@e...	5551122333	10.05.2023
32	İpek	An	5500	ipek.ari@exampl...	5552233444	5.04.2023
33	Emir	Uzun	6000	emir.uzun@exam...	5553344555	1.03.2023
34	Ayşe	Demir	6100	ayse.demir@exa...	5554455666	10.02.2023
35	Rıza	Toprak	5400	riza.toprak@exa...	5555566777	20.01.2023
21	Ece	Aydın	5200	ece.aydin@exam...	05551112233	1.03.2024

Resim 1.0.0 Arama

aranacak tablo butonuna basılır(örnekte personel)

%aranan%  
an  
FİLTRELE

Resim 1.0.1 Arama

aranacak kelimenin konumu ve kelime girilir filtrele butonuna basılır(örnekte an)

	personelId	personelAd	personelSoyad	sabitUcret	ePosta	telefonNo	girisTarih
▶	24	Caner	Bozkurt	6000	caner.bozkurt@e...	5554455666	10.12.2023
	31	Batuhan	Kaya	5700	batuhan.kaya@e...	5551122333	10.05.2023
	13	Hakan	Turan	6400	hakan.turan@ex...	5553333444	10.04.2023
	19	Can	Kılıç	5400	can.kilic@examp...	5559999000	20.10.2022
	41	Can	Toprak	5500	can.toprak@exa...	5556677888	15.12.2023
*							

**Resim 1.0.2 Arama**

*Sonuç tablosu listelenir*

## 2-EKLEME İŞLEMİ:

urunid	kategorid	tabaklamaid	kaynakid	yuzeyid	stokmiktarı	satısfıyatı	urunad
5	4	2	5	2	119	450	Deri Ayakkabı
6	2	3	6	1	48	750	Deri Koltuk ...
8	6	1	8	3	248	650	Deri Çeket
9	1	3	9	2	89	800	El Yapımı D...
10	3	2	10	1	128	350	Deri Portföy
11	2	1	1	4	59	850	Deri Etek
12	4	3	2	3	68	920	Deri Elbise
13	3	2	3	2	179	500	Deri Otomo...
14	5	1	4	4	108	450	Deri Ayakkabı
15	6	4	5	1	74	620	Deri Çanta ...
16	7	3	6	2	198	550	Süet Ayakk...
17	8	1	7	3	219	700	Süet Çeket
18	5	2	8	4	48	480	Deri El Çant...
19	4	3	9	1	179	500	Deri Sırt Ç...
20	2	1	10	2	158	600	Deri Portföy...
1	1	1	1	1	99	1200	Klasik Deri ...
2	7	3	4	4	100	123456	AYAKKABI
3	2	2	3	1	199	550	Deri Koltuk
4	3	1	4	3	178	700	Deri Çanta

Urun ID: 21

Urunun Adı: DENEME EKLEME

Kaynak Tür ID: Dana Derisi

Yüzey İşleme Tür ID: Pürüzsüz Deri

Kategori Tür ID: El İgi

Tabaklama Tür ID: Derin Tabaklama

Stok Miktarı: 2

Satış Fiyatı: 2

Tablolar

>>

Listele

Ekle

Sil

Güncelle

ÇIKIŞ

**Resim 2.0.0 Ekleme**

*Eklenecek ürün için uygun değerler girilir*

Ekle

**Resim 2.0.1 Ekleme**

*Ekle butonuna basılır*

urun ekleme işlemi başarılı bir şekilde gerçekleştir

Tamam



### Resim 2.0.2 Ekleme

*Ekleme başarılı uyarı ekranı görüntülenir*

4	3	1	4	3	1/8	700	Deri Çanta
21	1	1	1	1	2	2	DENEME EK...
*							

### Resim 2.0.3 Ekleme

*Listele butonuna basarak Ekleme sonucu kontrol edilebilir*

## 3-SİLME İŞLEMİ :

urunid	kategoriid	tabaklamaid	kaynakid	yuzeyid	stokmiktari	satisfiyati	urunad
5	4	2	5	2	119	450	Deri Ayakkabı
6	2	3	6	1	48	750	Deri Koltuk ...
8	6	1	8	3	248	650	Deri Ceket
9	1	3	9	2	89	800	El Yapımı D...
10	3	2	10	1	128	350	Deri Portföy
11	2	1	1	4	59	850	Deri Etek

Urun ID:

Urunun Adı:

Kaynak Tür ID:

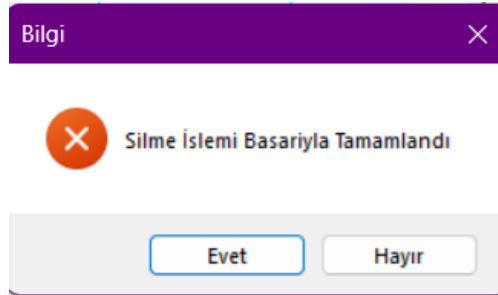
### Resim 3.0.0 Silme

*Silinecek id girilir*



### Resim 3.0.1 Silme

*Sil butonuna basılır*



### Resim 3.0.2 Silme

*Silme uyarı ekranı çıkar*

urunid	kategoriid	tabaklamaid	kaynakid	yuzeyid	stokmiktari	satisfiyati	urunad
6	2	3	6	1	48	750	Deri Koltuk ...
8	6	1	8	3	248	650	Deri Ceket
9	1	3	9	2	89	800	El Yapımı D...
10	3	2	10	1	128	350	Deri Portföy
11	2	1	1	4	59	850	Deri Etek
12	4	2	2	2	68	920	Deri Elbise

**Resim 3.0.3 Silme**

*Silme işlemi tablodan kontrol edilebilir*

#### 4-GÜNCELLEME İŞLEMİ

urunid	kategoriid	tabaklamaid	kaynakid	yuzeyid	stokmiktari	satisfiyati	urunad
6	2	3	6	1	48	750	Deri Koltuk ...

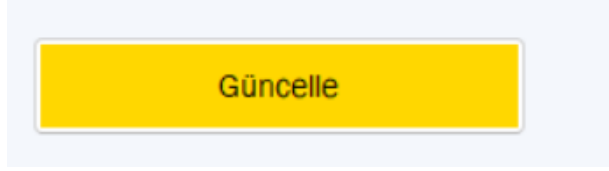
**Resim 4.0.0 Güncelleme**

*Güncellemeden önce verilerin görüntüsü*

Urun ID:	<input type="text" value="6"/>
Urunun Adı:	<input type="text" value="GÜNCELLENECEK"/>
Kaynak Tür ID:	<input type="text" value="Kuzu Derisi"/>
Yüzey İşleme Tür ID:	<input type="text" value="Boya Derisi"/>
Kategori Tür ID:	<input type="text" value="Yapı Malzemeleri"/>
Tabaklama Tür ID:	<input type="text" value="Sünger Tabaklama"/>
Stok Miktarı:	<input type="text" value="100"/>
Satış Fiyatı:	<input type="text" value="11"/>

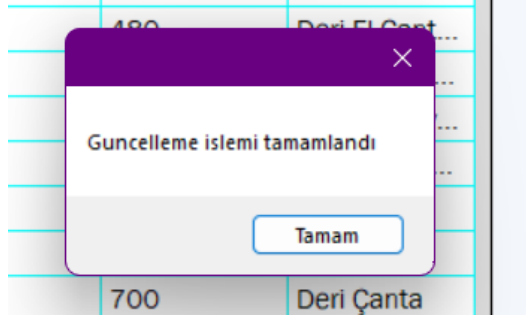
**Resim4.0.1 Güncelleme**

*Güncel veriler girilir ( id aynı olmalı )*



#### Resim4.0.2 Güncelleme

*Güncelle butonuna basılır*



#### Resim4.0.3 Güncelleme

*Güncelleme işlemi sonucu ile ilgili bilgi ekranı*

	6	6	4	4	4	100	11	GÜNCELLEN...
*								

#### Resim4.0.4 Güncelleme

*Güncellenmiş verilerin görüntüsü*