

VERİTABANI PROJESİ

-Serkan ÇİFCİ

S211210204

serkan_7578@hotmail.com

Ozan ÇAKMAK

s211210187

Email : cakmak.ozan42@gmail.com

GITHUB LINKI: https://github.com/csOzan/dbms_22summer

TANITIM:

Uygulamamız araç ticareti üzerinden gerçekleştirmekle birlikte kredi danışmanlığını da kapsamaktadır.

İŞ KURALLARI:

Her kişinin bir iletişim bilgisi bulunur.

Her bir kişinin kişi tipi olarak bir özelliği bulunur.

Kişi; satıcı,müşteri veya müşteri hizmetlerinden biri olabilir.

Satıcı birden fazla araç satabilir.

Müşteri birden fazla araç satın alabilir.

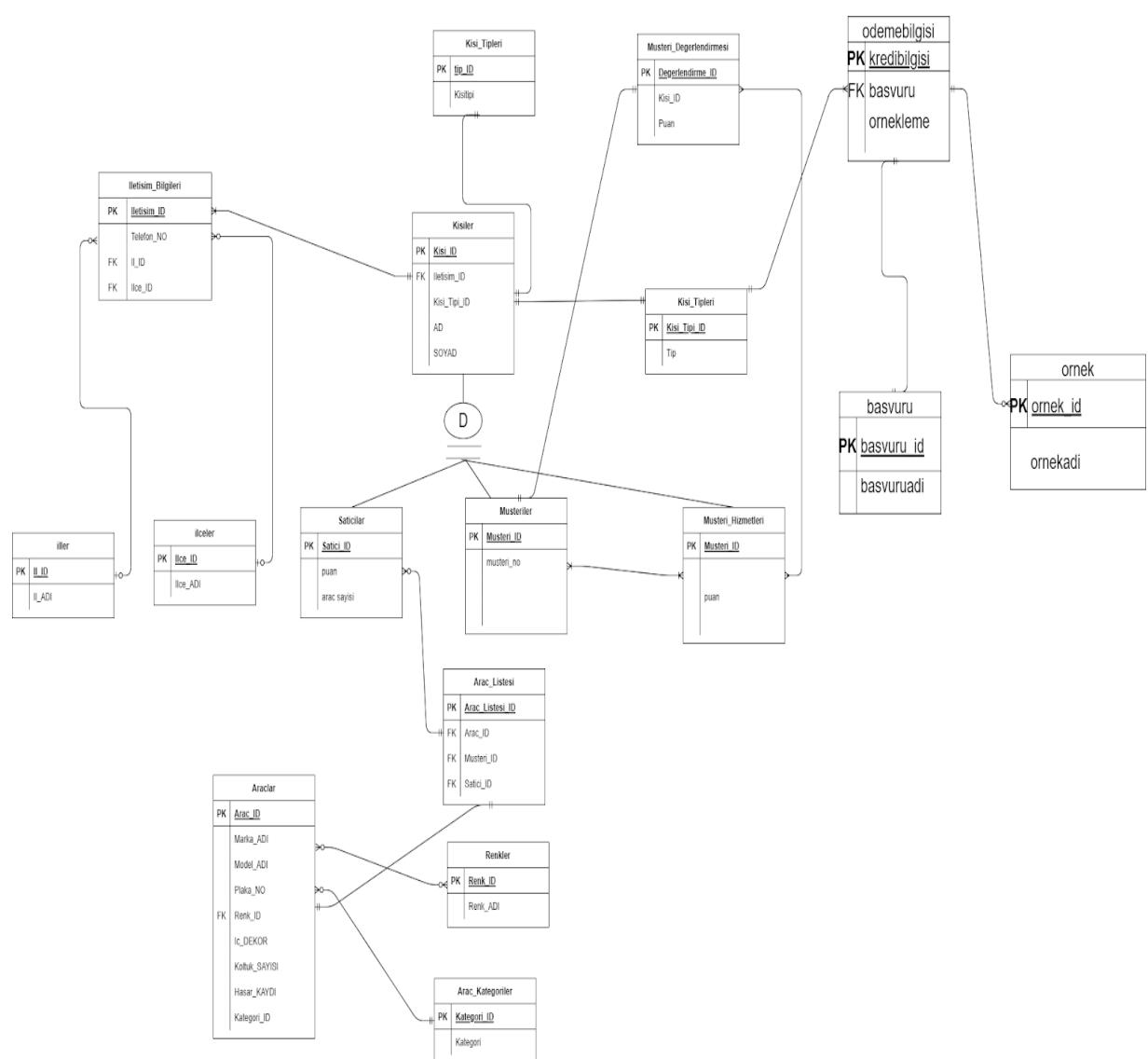
Bir müşteri hizmetlerinden kişi bir müşteriyle ilgilenebilir.

Müşteri hizmetleri kredi danışmanlığını sunar.

Aracın kategorileri, dış görünüm ve kullanım bilgileri yer alır.

Kredi danışmanlığında başvuru bilgisi ve örnekler yer alır.

Varlık Bağıntı modeli (Crow's Foot, Kalıtım)



SQL kodları:

```
1 -- Database: postgres
2
3 -- DROP DATABASE IF EXISTS postgres;
4
5 ✓ CREATE DATABASE postgres
6   |
7     OWNER = postgres
8     ENCODING = 'UTF8'
9     LC_COLLATE = 'Turkish_Turkey.utf8'
10    LC_CTYPE = 'Turkish_Turkey.utf8'
11    TABLESPACE = pg_default
12    CONNECTION LIMIT = -1
13    IS_TEMPLATE = False;
14
15 ✓ COMMENT ON DATABASE postgres
16   IS 'default administrative connection database';
17
18
19 -- Table: public.Arac_Listesi
20
21 -- DROP TABLE IF EXISTS public."Arac_Listesi";
22
23 CREATE TABLE IF NOT EXISTS public."Arac_Listesi"
24 ✓ (
25   "Arac_Listesi_ID" integer NOT NULL,
26   "Arac_ID" integer NOT NULL,
27   "Musteri_ID" integer NOT NULL,
28   "Satici_ID" integer NOT NULL,
29   CONSTRAINT "Arac_Listesi_pkey" PRIMARY KEY ("Arac_Listesi_ID")
30 )
31
32 TABLESPACE pg_default;
33
34 ✓ ALTER TABLE IF EXISTS public."Arac_Listesi"
35   OWNER to postgres;
36
```

```
38
39 -- DROP TABLE IF EXISTS public."Arac_Kategoriler";
40
41
42
43 CREATE TABLE IF NOT EXISTS public."Arac_Kategoriler"
44 (
45   "Kategori_ID" integer NOT NULL,
46   "Kategori" character varying COLLATE pg_catalog."default" NOT NULL,
47   CONSTRAINT "Arac_Kategoriler_pkey" PRIMARY KEY ("Kategori_ID")
48 )
49
50 TABLESPACE pg_default;
51
52
53
54 ALTER TABLE IF EXISTS public."Arac_Kategoriler"
55   OWNER to postgres;
56
57   -- Table: public.Araclar
58
59 -- DROP TABLE IF EXISTS public."Araclar";
60
61 CREATE TABLE IF NOT EXISTS public."Araclar"
62 (
63   "Arac_ID" integer NOT NULL,
64   "Marka_ADI" character varying COLLATE pg_catalog."default" NOT NULL,
65   "Model_ADI" character varying COLLATE pg_catalog."default" NOT NULL,
66   "Plaka_NO" integer NOT NULL,
67   "Renk_ID" integer NOT NULL,
68   "Ic_DEKOR" character varying COLLATE pg_catalog."default" NOT NULL,
69   "Koltuk_SAYISI" integer NOT NULL,
70   "Hasar_KAYIDI" character varying COLLATE pg_catalog."default" NOT NULL,
71   "Kategori_ID" integer NOT NULL,
72   CONSTRAINT "Araclar_pkey" PRIMARY KEY ("Arac_ID")
```

```
74
75   TABLESPACE pg_default;
76
77   ✓ ALTER TABLE IF EXISTS public."Araclar"
78     OWNER to postgres;
79
80   -- Table: public.Kisi_tipleri
81
82   ✓
83   -- DROP TABLE IF EXISTS public."Kisi_tipleri";
84
85   CREATE TABLE IF NOT EXISTS public."Kisi_tipleri"
86   ✓ (
87     "tip_ID" integer NOT NULL,
88     kisitipi character varying COLLATE pg_catalog."default" NOT NULL,
89     CONSTRAINT "Kisi_tipleri_pkey" PRIMARY KEY ("tip_ID")
90   )
91
92   TABLESPACE pg_default;
93
94   ✓ ALTER TABLE IF EXISTS public."Kisi_tipleri"
95     OWNER to postgres;
96
97   -- Table: public.Musteri_Degerlendirmesi
98
99   -- DROP TABLE IF EXISTS public."Musteri_Degerlendirmesi";
100
101  CREATE TABLE IF NOT EXISTS public."Musteri_Degerlendirmesi"
102  ✓ (
103    "Degerlendirme_ID" integer NOT NULL,
104    "Kisi_ID" integer NOT NULL,
105    "Puan" integer NOT NULL,
106    CONSTRAINT "Musteri_Degerlendirmesi_pkey" PRIMARY KEY ("Degerlendirme_ID")
107  )
108
109  TABLESPACE pg_default;
```

```
111 ✓ ALTER TABLE IF EXISTS public."Musteri_Degerlendirmesi"
112     OWNER to postgres;
113
114
115     -- Table: public.Musteri_Hizmetleri
116
117 -- DROP TABLE IF EXISTS public."Musteri_Hizmetleri";
118
119 CREATE TABLE IF NOT EXISTS public."Musteri_Hizmetleri"
120 (
121     -- Inherited from table public.kisiler: "Kisi_ID" integer NOT NULL,
122     -- Inherited from table public.kisiler: "Iletisim_ID" integer NOT NULL,
123     -- Inherited from table public.kisiler: "Kisi_tipi_ID" integer NOT NULL,
124     -- Inherited from table public.kisiler: "AD" character varying COLLATE pg_catalog."default" NOT NULL,
125     -- Inherited from table public.kisiler: "SOYAD" character varying COLLATE pg_catalog."default" NOT NULL,
126     "Musteri_Hizmetleri_ID" integer NOT NULL,
127     puan integer NOT NULL,
128     CONSTRAINT "Musteri_Hizmetleri_pkey" PRIMARY KEY ("Musteri_Hizmetleri_ID")
129 )
130 INHERITS (public.kisiler)
131 TABLESPACE pg_default;
132
133 ✓ ALTER TABLE IF EXISTS public."Musteri_Hizmetleri"
134     OWNER to postgres;
135
136     -- Table: public.Musteriler
137
138 -- DROP TABLE IF EXISTS public."Musteriler";
139
140 CREATE TABLE IF NOT EXISTS public."Musteriler"
141 (
142     -- Inherited from table public.kisiler: "Kisi_ID" integer NOT NULL,
143     -- Inherited from table public.kisiler: "Iletisim_ID" integer NOT NULL,
144     -- Inherited from table public.kisiler: "Kisi_tipi_ID" integer NOT NULL,
145     -- Inherited from table public.kisiler: "AD" character varying COLLATE pg_catalog."default" NOT NULL,
```

```
154 ✓ ALTER TABLE IF EXISTS public."Musteriler"
155   OWNER to postgres;
156
157   -- Table: public.Renkler
158
159   -- DROP TABLE IF EXISTS public."Renkler";
160
161 CREATE TABLE IF NOT EXISTS public."Renkler"
162 ✓(
163   "Renk_ID" integer NOT NULL,
164   "Renk_ADI" character varying COLLATE pg_catalog."default" NOT NULL,
165   CONSTRAINT "Renkler_pkey" PRIMARY KEY ("Renk_ID")
166 )
167
168 TABLESPACE pg_default;
169
170 ✓ ALTER TABLE IF EXISTS public."Renkler"
171   OWNER to postgres;
172
173
174   -- DROP TABLE IF EXISTS public."Saticilar";
175
176 CREATE TABLE IF NOT EXISTS public."Saticilar"
177 ✓(
178   -- Inherited from table public.kisiler: "Kisi_ID" integer NOT NULL,
179   -- Inherited from table public.kisiler: "Iletisim_ID" integer NOT NULL,
180   -- Inherited from table public.kisiler: "Kisi_tipi_ID" integer NOT NULL,
181   -- Inherited from table public.kisiler: "AD" character varying COLLATE pg_catalog."default" NOT NULL,
182   -- Inherited from table public.kisiler: "SOYAD" character varying COLLATE pg_catalog."default" NOT NULL,
183   "Satici_ID" integer NOT NULL,
184   puan integer NOT NULL,
185   arac_sayisi integer NOT NULL,
```

```
191 ✓ ALTER TABLE IF EXISTS public."Saticilar"
192   OWNER to postgres;
193
194
195   -- Table: public.basvuru
196
197   -- DROP TABLE IF EXISTS public.basvuru;
198
199 CREATE TABLE IF NOT EXISTS public.basvuru
200 ✓(
201   "basvuru_ID" integer NOT NULL,
202   basvuruadi character varying COLLATE pg_catalog."default",
203   CONSTRAINT basvuru_pkey PRIMARY KEY ("basvuru_ID")
204 )
205
206
207 TABLESPACE pg_default;
208
209 ✓ ALTER TABLE IF EXISTS public.basvuru
210   OWNER to postgres;
211
212
213   -- Table: public.ilceler
214
215   -- DROP TABLE IF EXISTS public.ilceler;
216
217 CREATE TABLE IF NOT EXISTS public.ilceler
218 ✓(
219   "ilce_ID" integer NOT NULL,
220   "ilce_ADI" character varying COLLATE pg_catalog."default" NOT NULL,
221   CONSTRAINT ilceler_pkey PRIMARY KEY ("ilce_ID")
222 )
223
224 TABLESPACE pg_default;
```

```
262 ✓ ALTER TABLE IF EXISTS public.iller
263     OWNER to postgres;
264
265     -- Table: public.kisiler
266
267 -- DROP TABLE IF EXISTS public.kisiler;
268
269 CREATE TABLE IF NOT EXISTS public.kisiler
270  (
271     "Kisi_ID" integer NOT NULL,
272     "Iletisim_ID" integer NOT NULL,
273     "Kisi_tipi_ID" integer NOT NULL,
274     "AD" character varying COLLATE pg_catalog."default" NOT NULL,
275     "SOYAD" character varying COLLATE pg_catalog."default" NOT NULL,
276     CONSTRAINT kisiler_pkey PRIMARY KEY ("Kisi_ID")
277 )
278
279 TABLESPACE pg_default;
280
281 ✓ ALTER TABLE IF EXISTS public.kisiler
282     OWNER to postgres;
283
284     -- Table: public.odemebilgisi
285
286 -- DROP TABLE IF EXISTS public.odemebilgisi;
287
288 CREATE TABLE IF NOT EXISTS public.odemebilgisi
289  (
290     kredibilgisi character varying COLLATE pg_catalog."default" NOT NULL,
291     basvuru character varying COLLATE pg_catalog."default" NOT NULL,
292     ornekleme character varying COLLATE pg_catalog."default" NOT NULL,
293     CONSTRAINT odemebilgisi_pkey PRIMARY KEY (kredibilgisi)
294 )
295
```

```

298 ✓ ALTER TABLE IF EXISTS public.odemebilgisi
299   OWNER to postgres;
300
301
302   -- Table: public.ornek
303
304   -- DROP TABLE IF EXISTS public.ornek;
305
306 CREATE TABLE IF NOT EXISTS public.ornek
307 ✓ (
308   "ornek_ID" integer NOT NULL,
309   ornekadi character varying COLLATE pg_catalog."default" NOT NULL,
310   CONSTRAINT ornek_pkey PRIMARY KEY ("ornek_ID")
311 )
312
313 TABLESPACE pg_default;
314
315 ✓ ALTER TABLE IF EXISTS public.ornek
316   | OWNER to postgres;

```

Tetikleyici

1) Araç bilgisi:

```

CREATE TABLE "public"."AracDegisikligilzle" (
    "kayitNo" serial,
    "urunNo" SmallInt NOT NULL,
    "eskiBirimFiyat" Real NOT NULL,
    "yeniBirimFiyat" Real NOT NULL,
    "degisiklikTarihi" TIMESTAMP NOT NULL,
    CONSTRAINT "PK" PRIMARY KEY ("kayitNo")
);
CREATE OR REPLACE FUNCTION "urunDegisikligiTR1"()
RETURNS TRIGGER
AS
$$
BEGIN
    IF NEW."UnitPrice" <> OLD."UnitPrice" THEN
        INSERT INTO "UrunDegisikligilzle"("urunNo", "eskiBirimFiyat", "yeniBirimFiyat",
        "degisiklikTarihi")

```

```

        VALUES(OLD."ProductID", OLD."UnitPrice", NEW."UnitPrice",
CURRENT_TIMESTAMP::TIMESTAMP);
END IF;

RETURN NEW;
END;
$$
LANGUAGE "plpgsql";
[ CREATE TRIGGER "urunBirimFiyatDegistiginde"
BEFORE UPDATE ON "products"
FOR EACH ROW
EXECUTE PROCEDURE "urunDegisikligiTR1"();
UPDATE "products"
SET "UnitPrice" = 100
WHERE "ProductID" = 4

```

2-3) ÜRÜN İÇERİĞİ DEĞİŞİKLİĞİ VE GÜNCELLEME:

[00:40, 28.08.2022] Serkan: CREATE OR REPLACE FUNCTION "kayitEkleTR1"()
 RETURNS TRIGGER
 AS
 \$\$
 BEGIN
 NEW."CompanyName" = UPPER(NEW."CompanyName"); -- büyük harfe dönüştürdükten
 sonra ekle
 NEW."ContactName" = LTRIM(NEW."ContactName"); -- Önceki ve sonraki boşlukları
 temizle
 IF NEW."City" IS NULL THEN
 RAISE EXCEPTION 'Sehir alanı boş olamaz';
 END IF;
 RETURN NEW;
 END;
 \$\$
 LANGUAGE "plpgsql";
CREATE TRIGGER "kayitKontrol"
BEFORE INSERT OR UPDATE ON "customers" -- veriyi eklemeden/değiştirmeden önce

```

üzerinde işlem yap
FOR EACH ROW
EXECUTE PROCEDURE "kayitEkleTR1"();
INSERT INTO "customers" ( "CustomerID","CompanyName", "ContactName")
VALUES ( '45', 'Orka Ltd.', ' Ayşe Yalın ' );
INSERT INTO "customers" ( "CustomerID","CompanyName", "ContactName", "City")
ALTER TABLE "products"
DISABLE TRIGGER "urunBirimFiyatDegistiginde";
ALTER TABLE "products"
ENABLE TRIGGER "urunBirimFiyatDegistiginde";
ALTER TABLE "products"
DISABLE TRIGGER ALL;
ALTER TABLE "products"
ENABLE TRIGGER ALL;
DROP TRIGGER "urunBirimFiyatDegistiginde" ON "products";
DROP TRIGGER IF EXISTS "urunBirimFiyatDegistiginde" ON "products";

```

4) Yeni tablo oluşturmak için:

```

DROP TABLE IF EXISTS araclar;

CREATE TABLE araclar(
    id INT GENERATED ALWAYS AS IDENTITY,
    first_name VARCHAR(40) NOT NULL,
    last_name VARCHAR(40) NOT NULL,
    PRIMARY KEY(id)
);
CREATE TABLE aracbilgisi(
    id INT GENERATED ALWAYS AS IDENTITY,
    employee_id INT NOT NULL,
    last_name VARCHAR(40) NOT NULL,
    changed_on TIMESTAMP(6) NOT NULL
);
CREATE OR REPLACE FUNCTION log_last_name_changes()
RETURNS TRIGGER
LANGUAGE PLPGSQL
AS
$$
BEGIN

```

```

IF NEW.last_name <> OLD.last_name THEN
    INSERT INTO aracbilgisi(arac_id,last_name,changed_on)
        VALUES(OLD.id,OLD.last_name,now());
END IF;

RETURN NEW;
END;
$$
CREATE TRIGGER last_name_changes
BEFORE UPDATE
ON araclar
FOR EACH ROW
EXECUTE PROCEDURE log_last_name_changes();
INSERT INTO araclar (first_name, last_name)
VALUES (arac1, arac2);

INSERT INTO arac (first_name, last_name)
VALUES (arac3, arac4);
SELECT * FROM araclar;
UPDATE araclar
SET last_name = 'Brown'
WHERE ID = 2;
SELECT * FROM arac;
SELECT * FROM aracbilgisi;

```

Saklı Yordamlar

Proje için 4 farklı saklı yordam yazdım. Bunları sırasıyla başlıklar altında aşağıda açıklayacağım. Açıklamalarım ve ekran görüntülerim aynı zamanda Githubdaki readme dosyalarında bulunmaktadır. (Bu kısım için /sql/readme.md bakabilirsiniz)

int MusteriSayisi(void):

MusteriSayısı fonksiyonu müşteriler tablosundaki müşteri sayısını integer olarak döner. Bunu Count fonksiyonu yardımıyla yapar. API de kullanmak için “musterisayisi” strinigini kullanabilirsiniz

```
1 CREATE OR REPLACE FUNCTION MusteriSayisi()
2 RETURNS INTEGER AS $total$
3 declare
4     total integer;
5 ▼ BEGIN
6     SELECT count(*) into total FROM musteriler;
7     RETURN total;
8 END;
9 $total$ LANGUAGE plpgsql;
10

dbms22_summer=# Select * From musteriler;
 Kisi_ID | Iletisim_ID | Kisi_tipi_ID |      AD       | SOYAD    | Musteri_ID | kredi_karti
-----+-----+-----+-----+-----+-----+-----+
  6 |           |        2 | Refik Halit | Karay    |      1 | 98385928
  7 |           |        2 | Halide Edip | Adivar   |      2 | 74379246
  8 |           |        2 | Cemil        | Suleyman |      3 | 47268463
  9 |           |        2 | Ahmet        | Mithat   |      4 | 56735947
 10 |          |        2 | Ziya         | Gokalp   |      5 | 23122321
(5 rows)

dbms22_summer=# Select MusteriSayisi();
musterisayisi
-----
      5
(1 row)

dbms22_summer=#
```

int KisiSayisi(void):

KisiSayısı fonksiyonu, kisiler tablosunu kalıtım olarak alan tabloların (müşteri_hizmetleri, musteriler, satıcılar) toplam satır sayısını döner. Burada count fonksiyondan faydalanyılır. API’de kullanmak için “kisisayisi” strinigini kullanabilirsiniz.

```
10
11 CREATE OR REPLACE FUNCTION KisiSayisi()
12 RETURNS INTEGER AS $total$
13 declare
14     total integer;
15 ▼ BEGIN
16     SELECT count(*) into total FROM kisiler;
17     RETURN total;
18 END;
19 $total$ LANGUAGE plpgsql;
20
```

```
dbms22_summer=# Select Count(*) From musteri_hizmetleri;
count
-----
      6
(1 row)

dbms22_summer=# Select Count(*) From musteriler;
count
-----
      5
(1 row)

dbms22_summer=# Select Count(*) From staticilar;
count
-----
      5
(1 row)

dbms22_summer=# Select KisiSayisi();
kisisayisi
-----
      16
(1 row)

dbms22_summer=# []
```

void IISil (int kod):

Argüman olarak verilen Id'li İli iller tablosundan siler. API'de kullanmak için "ilsil" strinigini kullanabilirsiniz.

```
21 -- Arguman olarak verilen koddaki ili silmeyi saglayan fonksiyon
22 CREATE OR REPLACE FUNCTION Ilsil (kod integer) RETURNS void AS $$ 
23 ▼ BEGIN
24     DELETE FROM iller WHERE "il_ID"=kod;
25 END;
26 $$ LANGUAGE plpgsql;
```

```
dbms22_summer=# Select * From iller;
 il_ID | il_adi
-----+-----
    1 | Antalya
    2 | Ankara
    3 | Samsun
    4 | Bolu
    5 | Sakarya
    6 | İstanbul
(6 rows)

dbms22_summer=# Select Ilsil(3);
ilsil
-----
(1 row)

dbms22_summer=# Select * From iller;
 il_ID | il_adi
-----+-----
    1 | Antalya
    2 | Ankara
    4 | Bolu
    5 | Sakarya
    6 | İstanbul
(5 rows)

dbms22_summer=#
```

void SaticiMusteriHizSil (void):

Müşteri hizmetleri ve satıcılar tablosundan puanı 5'in altındaki kişileri siler.
API'de kullanmak için "sil" strinigini kullanabilirsiniz.

```

27
28 -- Puanı 5'in altında olan satıcı ve müşteri hizmetlerinin silinmesini sağlayan fonksiyon
29 CREATE OR REPLACE FUNCTION SaticiMusteriHizSil() RETURNS void AS $$ 
30 ▼ BEGIN
31     DELETE FROM satıcılar WHERE "puan" < 5;
32     DELETE FROM müşteri_hizmetleri WHERE "puan" < 5;
33 END;
34 $$ LANGUAGE plpgsql;

```

```

dbms22_summer=# Select * From müşteri_hizmetleri;
Kisi_ID | Iletisim_ID | Kisi_tipi_ID | AD | SOYAD | Musteri_Hizmetleri_ID | puan
-----+-----+-----+-----+-----+-----+-----+
    13 |           |         3 | Demir | Demirkan |             3 |    7
    15 |           |         3 | Ogun  | Sanlisoy |             5 |    6
    17 |           |         3 | Hayko | Cepkin  |             7 |   10
    18 |           |         3 | Sebnem | Ferah   |             8 |    6
    14 |           |         3 | Fatma | Turgut  |             4 |    3
    16 |           |         3 | Ferman | Akgül   |             6 |    2
(6 rows)

dbms22_summer=# Select * From satıcılar;
Kisi_ID | Iletisim_ID | Kisi_tipi_ID | AD | SOYAD | Satici_ID | puan | arac_sayisi
-----+-----+-----+-----+-----+-----+-----+-----+
     2 |           |         1 | Zaim  | Gurdamar |          2 |    8 |      10
     3 |           |         1 | Hasim | Gurdamar |          3 |    9 |       7
     5 |           |         1 | Muhammed | Zaim   |          5 |    6 |       3
    19 |           |         1 | Ilhan | Ozdemir |          6 |    2 |      20
    20 |           |         1 | Furkan | Deniz   |          7 |    3 |      14
(5 rows)

dbms22_summer=# Select SaticiMusteriHizSil();
saticimusterihizsil
-----
(1 row)

dbms22_summer=# Select * From satıcılar;
Kisi_ID | Iletisim_ID | Kisi_tipi_ID | AD | SOYAD | Satici_ID | puan | arac_sayisi
-----+-----+-----+-----+-----+-----+-----+-----+
     2 |           |         1 | Zaim  | Gurdamar |          2 |    8 |      10
     3 |           |         1 | Hasim | Gurdamar |          3 |    9 |       7
     5 |           |         1 | Muhammed | Zaim   |          5 |    6 |       3
(3 rows)

dbms22_summer=# Select * From müşteri_hizmetleri;
Kisi_ID | Iletisim_ID | Kisi_tipi_ID | AD | SOYAD | Musteri_Hizmetleri_ID | puan
-----+-----+-----+-----+-----+-----+-----+
    13 |           |         3 | Demir | Demirkan |             3 |    7
    15 |           |         3 | Ogun  | Sanlisoy |             5 |    6
    17 |           |         3 | Hayko | Cepkin  |             7 |   10
    18 |           |         3 | Sebnem | Ferah   |             8 |    6
(4 rows)

dbms22_summer=#

```

Arama, Ekleme, Silme, Güncelleme İşlemleri (CRUD) ve API

Uygulamada CRUD operasyonları yapılabilmesi için bir RESTfull API'si yazdım. Bunu NodeJS ve ExpressJS kullanarak JavaScript ile yazdım. Uygulama kaynak kodlarının içerisindeki **index.js**'nin **node** komutuna birincil argüman olarak gönderilmesi ile çalışır. Node komutu nodejs serverini çalıştırılmakta kullanılan bir komut satırı uygulamasıdır. API'nin genel olarak kullanımı şöyledir:

/ipofserver:5000 tablename

- Buraya POST ile erişerek istediğiniz tabloya satır ekleyebilirsiniz. Eklemek istediğiniz verileri json formatında göndermeniz gereklidir.
- Buraya GET ile erişerek istediğiniz tablonun içeriklerini sorgulayabilirsiniz.

/ipofserver:5000 tablename/:id

- Buraya GET ile erişerek ve istediğiniz ID yi vermeniz koşuluyla istediğiniz tablonun belirli ID'li satırına ulaşabilirsiniz
- Buraya PUT ile erişerek istediğiniz tablonun istediğiniz satırında güncelleme yapabilirsiniz. Verilerinizi json formatında göndermeniz gerekmektedir
- Buraya DELETE ile erişirseniz istediğiniz tablonun istediğiniz satırını silebilirsiniz

/ipofserver:5000/f/functionname

- Buradan seçtiğiniz gömülü fonksiyonu kullanabilirsiniz
- Kullanacağınız fonksiyon argüman kabul ediyorsa bunu {"argument": "yourargument"} formatında json olarak göndermeniz gerekmektedir
- functionname yerine yazılacak string'i fonksiyon tanıtımlarında görebilirsiniz.

Açıklamalarım ve ekran görüntülerim aynı zamanda Githubdaki readme dosyalarında bulunmaktadır. (Bu kısım için /api/readme.md ye bakabilirsiniz). Örnek CRUD işlemleri:

GET

The screenshot shows the Postman interface with a request configuration and a response preview.

Request Configuration:

- Method: GET
- URL: http://localhost:5000/ller/
- Headers:
 - Content-type: application/json

Response Preview (Pretty JSON):

```
1  [
2    {
3  "ller_id": 1,
4  "ller_ad": "Antalya"
5  },
6  {
7  "ller_id": 2,
8  "ller_ad": "Ankara"
9  },
10  {
11  "ller_id": 4,
12  "ller_ad": "Bolu"
13  },
14  {
15  "ller_id": 5,
16  "ller_ad": "Sakarya"
```

POST

The screenshot shows the Postman interface. In the left sidebar, under 'dbms22_summer' environment, there is a collection named 'iller' containing four items: 'iller_get', 'iller_post', 'iller_delete', and 'iller_put'. The 'iller_post' item is selected. The main panel shows a POST request to 'http://localhost:5000/iller'. The 'Body' tab is selected, showing a JSON payload: { "description": "Izmir" }. The response status is 200 OK, time 96 ms, size 373 B. The response body is: { "il_ID": 7, "il_adi": "Izmir" }.

GET AFTER POST

The screenshot shows the Postman interface. In the left sidebar, under 'dbms22_summer' environment, there is a collection named 'iller' containing four items: 'iller_get', 'iller_post', 'iller_delete', and 'iller_put'. The 'iller_get' item is selected. The main panel shows a GET request to 'http://localhost:5000/iller'. The 'Headers' tab is selected, with 'Content-type' set to 'application/json'. The response status is 200 OK, time 46 ms, size 529 B. The response body is a JSON array of city objects:

```

[{"il_ID": 4, "il_adi": "Bolu"}, {"il_ID": 5, "il_adi": "Sakarya"}, {"il_ID": 6, "il_adi": "Istanbul"}, {"il_ID": 7, "il_adi": "Izmir"}]

```

PUT

Postman interface showing a PUT request to `http://localhost:5000/iller/iller_put` with JSON body:

```

1: {
  "1": "Adana"
}

```

Response status: 200 OK

GET AFTER PUT

Postman interface showing a GET request to `http://localhost:5000/iller/iller_get` with Headers:

KEY	VALUE	DESCRIPTION
<code>Content-type</code>	<code>application/json</code>	

Response status: 200 OK

```

7: {
  "1": "Ankara"
}
8: {
  "1": "Sakarya"
}
9: {
  "1": "Istanbul"
}
10: {
  "1": "Izmir"
}
11: {
  "1": "Adana"
}
12: {
  "1": "Gaziantep"
}

```

DELETE

The screenshot shows the Postman application interface. On the left, the 'Scratch Pad' sidebar lists collections: 'dbms22_summer' (selected) and 'pos'. Under 'dbms22_summer', there are five requests: 'iller_get', 'iller_post', 'iller_delete' (highlighted in red), and 'iller_put'. The main workspace displays a 'DELETE' request to 'http://localhost:5000/iller/1'. The 'Headers' tab is selected, showing a single header 'Content-Type' with the value 'application/json'. The 'Body' tab is empty. The 'Test Results' tab shows a response status of '200 OK' with a time of '72 ms' and a size of '357 B'. The response body contains the string '1 "I1 Silindi!"'. A terminal window on the right shows the command 'node index.js' running on port 3000.

GET AFTER DELETE

This screenshot shows the same Postman setup as the previous one. The 'dbms22_summer' collection is still selected. The 'iller_get' request is now highlighted in green. The 'Headers' tab shows a 'Content-type' header with the value 'application/json'. The 'Body' tab is empty. The 'Test Results' tab shows a response status of '200 OK' with a time of '75 ms' and a size of '498 B'. The response body is a JSON array containing four objects, each representing a city: Ankara, Bolu, Sakarya, and Istanbul. The 'iller_delete' request from the previous screenshot has been executed, as evidenced by the successful 'iller_get' response.