BSM211-Proje Rapor Dosyası

Öğrenci No	B231210565
Öğrenci Adı ve Soyadı	Elvin Valiyev
Şube	1.A
E-Posta	elvin.valiyev@ogr.sakarya.edu.tr

	Değerlendirme Kriterleri	
1	İş Kuralları	
2	Crow s Foot Gösterimi	
3	İlişkisel Model-Metinsel Gösterim	
4	4 Tetikleyici	
5	4 Saklı Yordam ve ya Fonksiyon	
6	Kalıtım	
7	Yazılım Gerçekleme	
8	Rapor(Github linkide ilave olarak koyula bilir)	
9	SQL ifadeleri	
10	15 Tablo	

▼ Uygulama Kısa Tanıtım

-Uygulama SBB Toplu Taşıma hizmetlerini online ortamdada Yolculara ve Sürücülere ulaştırmayı hedefliyor, Kullanıcıların ve Şoförlerin istedikleri zaman Seferlerle ilgili bilgi ala bilmesi, Yolcuların sadece bir kaç dakikada oturdukları yerden bilet ala bilmesi, Durakları görüntüleye bilmesi, Yorum yapma ve Sürücü puanlama gibi özelliklere kullanıcı kısmında daha sık geribildirim almayı amaçlıyor. Sürücülerde ister otobüs ister minibüs olsun araçlarını sisteme kaydede bilir, hangi araçların sefere çıkacağını öğrene bilir ayrıca Araçların bakım durumlarınıda hiç bir yere gitmeden kontrol ede bilirler.

▼ İş Kuralları

1. Bir araç varsa ya o minibüstür ya da otobüstür, minibüs ve ya otobüs dışında araç var olamaz.
2. Bir araç aynı anda hem otobüs hem de minibüs olamaz.
3. Bir araç en çok çok sayıda sefere çıka bilir(en az hiç sefere çıkmaz)

4. Bir sefer varsa mutlaka ve mutlaka sadece 1 araç o seferi yapıyor demektir.
5. Bir aracın en çok çok sayıda bakım kaydı ola bilir (en az hiç olmaz)
6. Bir bakım kaydı varsa mutlaka ve mutlaka o kayıt sadece 1 araca aittir.
7. Bir sürücü varsa ya o minibüs sürücüsüdür ya da otobüs sürücüsüdür, minibüs sürücüsü ve ya otobüs sürücüsü dışında sürücü var olamaz.
8. Bir sürücü aynı anda hem otobüs sürücüsü hem de minibüs sürücüsü olamaz.
9. Bir sürücü en çok çok sayıda sefere çıka bilir(en az hiç sefere çıkmaz)
10. Bir sefer varsa mutlaka ve mutlaka sadece 1 sürücü o seferi yapıyor demektir.
11. Bir sürücüye ait en çok çok sayıda puanlama yapılmış ola bilir(en az hiç yapılmaz)
12. Bir puanlama yapılmışsa bu puanlama mutlaka ve mutlaka sadece 1 sürücüye ait ola bilir
13. Bir sefere ait en çok çok sayıda bilet satıla bilir(en az hiç satılmaz)
14. Bir bilet varsa o bilet sadece ve sadece 1 sefere aittir.
15. Bir durak hakkında en çok çok sayıda yorum yapıla bilir.(en az hiç yapılmaz)
16. Bir yorum yapılmışsa bu yorum sadece ve sadece 1 durağa ait ola bilir.
17. Bir yolcu en çok çok sayıda yorum yapa bilir(en az hiç yapmaz)
18. Bir yorum yapılmışsa bu yorum sadece ve sadece 1 yolcuya ait ola bilir.
19. Bir yolcu en çok çok sayıda şikayette buluna bilir(en az hiç bulunmaz)
20. Bir şikayette bulunullmuşsa bu şikayet sadece ve sadece 1 yolcuya ait ola bilir.
21. Bir bilet ile ilgili en çok çok sayıda ödeme yapılmıştır(en az 1 ödeme yapılmıştır)
22. 1 ödeme sadece ve sadece bir bilete ait ola bilir.
23. Bir yolcu en çok çok sayıda bilet almış ola bilir.
24. Bir bilet varsa bu bilet sadece ve sadece 1 yolcuya aittir.

▼ İlişkisel Model-Metinsel Gösterim

1. Araclar(aracID: SERIAL PK, plakaNO: VARCHAR(10), kapasite: INTEGER, surucuKategorisi: VARCHAR, bakimDurumu: BOOLEAN)
2. Suruculer(surucuID: SERIAL PK, ad: VARCHAR, soyad: VARCHAR, surucuKategorisi: VARCHAR, ehliyetTipi: VARCHAR)
3. Guzergahlar(guzergahID: SERIAL PK, isim: VARCHAR, baslangic: VARCHAR, bitis: VARCHAR)
4. Yolcular(yolcuID: SERIAL PK, ad: VARCHAR, soyad: VARCHAR, telefonNO: VARCHAR(11), email: VARCHAR)
5. Otobusler(aracID: INTEGER PK FK → Araclar.aracID, otobusTuru: VARCHAR)
6. Minibusler(aracID: INTEGER PK FK → Araclar.aracID, minibusTuru: VARCHAR)

7. OtobusSurucusu(surucuID: INTEGER PK FK → Suruculer.surucuID, otobusTecrubesi: INTEGER)
8. MinibusSurucusu(surucuID: INTEGER PK FK → Suruculer.surucuID, minibusTecrubesi: INTEGER)
9. Seferler(seferID: SERIAL PK, aracID: INTEGER FK → Araclar.aracID, surucuID: INTEGER FK → Suruculer.surucuID, guzergah: VARCHAR, tarih: DATE, saat: TIME)
10. AracModelleri(aracModelID: SERIAL PK, isim: VARCHAR, ozellikler: TEXT)
11. AracBakim(bakimID: SERIAL PK, aracID: INTEGER FK → Araclar.aracID, bakimTarihi: DATE, detaylar: TEXT)
12. SurucuPuani(puanID: SERIAL PK, surucuID: INTEGER FK → Suruculer.surucuID, puan: INTEGER, tarih: DATE)
13. Duraklar(durakID: SERIAL PK, isim: VARCHAR, konum: VARCHAR)
14. DurakYorumlari(yorumID: SERIAL PK, durakID: INTEGER FK → Duraklar.durakID, yolcuID: INTEGER FK → Yolcular.yolcuID, yorum: TEXT, tarih: DATE)
15. Sikayetler(sikayetID: SERIAL PK, yolcuID: INTEGER FK → Yolcular.yolcuID, konu: VARCHAR, detay: TEXT, tarih: DATE)
16. Biletler(biletID: SERIAL PK, seferID: INTEGER FK → Seferler.seferID, yolcuID: INTEGER FK → Yolcular.yolcuID, biletTutari: DECIMAL, satildigiTarih: DATE)
17. Promosyonlar(promosyonID: SERIAL PK, isim: VARCHAR, indirimOrani: DECIMAL, gecerlilikTarihi: DATE)
18. Odemeler(odemeID: SERIAL PK, biletID: INTEGER FK → Biletler.biletID, odemeTarihi: DATE, tutar: DECIMAL)

▼ SQL ifadeleri

Veritabanı oluşturma

```
CREATE DATABASE "sbbTopluTasima";
```

▼ Tablolar | 18 Tablo

1.Araclar

```
CREATE TABLE "Araclar"(
"aracID" SERIAL PRIMARY KEY,
"plakaNO" VARCHAR(10),
"kapasite" INTEGER,
"surucuKategorisi" VARCHAR,
"bakimDurumu" BOOLEAN
);
```

2.Suruculer

```
CREATE TABLE "Suruculer"(
"surucuID" SERIAL PRIMARY KEY,
"ad" VARCHAR,
"soyad" VARCHAR,
"surucuKategorisi" VARCHAR,
"ehliyetTipi" VARCHAR
);
```

3.Guzergahlar

```
CREATE TABLE "Guzergahlar"(
"guzergahID" SERIAL PRIMARY KEY,
"isim" VARCHAR,
"baslangic" VARCHAR,
"bitis" VARCHAR
);
```

4.Yolcular

```
CREATE TABLE "Yolcular"(
"yolcuID" SERIAL PRIMARY KEY,
"ad" VARCHAR,
"soyad" VARCHAR,
"telefonNO" VARCHAR(11),
"email" VARCHAR
);
```

5.Otobusler

```
CREATE TABLE "Otobusler"(
"aracID" INTEGER PRIMARY KEY,
"otobusTuru" VARCHAR,
CONSTRAINT "fkAracID" FOREIGN KEY("aracID") REFERENCES "Araclar"("aracI
D")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

6.Minibusler

```
CREATE TABLE "Minibusler"(
"aracID" INTEGER PRIMARY KEY,
"minibusTuru" VARCHAR,
CONSTRAINT "fkAracID" FOREIGN KEY("aracID") REFERENCES "Araclar"("aracI
D")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

7.OtobusSurucusu

```
CREATE TABLE "OtobusSurucusu"(
"surucuID" INTEGER PRIMARY KEY,
"otobusTecrubesi" INTEGER,
CONSTRAINT "fkSurucuID" FOREIGN KEY("surucuID") REFERENCES "Suruculer"
("surucuID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

8.MinibusSurucusu

```
CREATE TABLE "MinibusSurucusu"(
"surucuID" INTEGER PRIMARY KEY,
"minibusTecrubesi" INTEGER,
CONSTRAINT "fkSurucuID" FOREIGN KEY("surucuID") REFERENCES "Suruculer"
("surucuID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

10.Seferler

```
CREATE TABLE "Seferler"(
"seferID" SERIAL PRIMARY KEY,
"aracID" INTEGER,
"surucuID" INTEGER,
"guzergah" VARCHAR,
"tarih" DATE,
"saat" TIME,
CONSTRAINT "fkAracID" FOREIGN KEY("aracID") REFERENCES "Araclar"("aracI
D")
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT "fkSurucuID" FOREIGN KEY("surucuID") REFERENCES "Suruculer"
("surucuID")
```

```
ON DELETE CASCADE ON UPDATE CASCADE
);
```

11.AracModelleri

```
CREATE TABLE "AracModelleri"(
"aracModelID" SERIAL PRIMARY KEY,
"isim" VARCHAR,
"ozellikler" TEXT
);
```

12.AracBakim

```
CREATE TABLE "AracBakim"(
"bakimID" SERIAL PRIMARY KEY,
"aracID" INTEGER,
"bakimTarihi" DATE,
"detaylar" TEXT,
CONSTRAINT "fkAracID" FOREIGN KEY("aracID") REFERENCES "Araclar"("aracI
D")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

13.SurucuPuani

```
CREATE TABLE "SurucuPuani"(
"puanID" SERIAL PRIMARY KEY,
"surucuID" INTEGER,
"puan" INTEGER,
"tarih" DATE,
CONSTRAINT "fkSurucuID" FOREIGN KEY("surucuID") REFERENCES "Suruculer"
("surucuID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

14.Duraklar

```
CREATE TABLE "Duraklar"(
"durakID" SERIAL PRIMARY KEY,
"isim" VARCHAR,
```

```
"konum" VARCHAR
);
```

15.DurakYorumlari

```
CREATE TABLE "DurakYorumlari"(
"yorumID" SERIAL PRIMARY KEY,
"durakID" INTEGER,
"yolcuID" INTEGER,
"yorum" TEXT,
"tarih" DATE,
CONSTRAINT "fkDurakID" FOREIGN KEY("durakID") REFERENCES "Duraklar"("du
rakID")
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT "fkYolcuID" FOREIGN KEY("yolcuID") REFERENCES "Yolcular"("yo
lcuID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

16.Sikayetler

```
CREATE TABLE "Sikayetler"(
"sikayetID" SERIAL PRIMARY KEY,
"yolcuID" INTEGER,
"konu" VARCHAR,
"detay" TEXT,
"tarih" DATE,
CONSTRAINT "fkYolcuID" FOREIGN KEY("yolcuID") REFERENCES "Yolcular"("yo
lcuID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

17.Biletler

```
CREATE TABLE "Biletler"(
"biletID" SERIAL PRIMARY KEY,
"seferID" INTEGER,
"yolcuID" INTEGER,
"biletTutari" DECIMAL,
"satildigiTarih" DATE,
CONSTRAINT "fkSeferID" FOREIGN KEY("seferID") REFERENCES "Seferler"("se
ferID")
```

```
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT "fkYolcuID" FOREIGN KEY("yolcuID") REFERENCES "Yolcular"("yo
lcuID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

18.Promosyonlar

```
CREATE TABLE "Promosyonlar"(
"promosyonID" SERIAL PRIMARY KEY,
"isim" VARCHAR,
"indirimOrani" DECIMAL,
"gecerlilikTarihi" DATE
);
```

Odemeler

```
CREATE TABLE "Odemeler"(
"odemeID" SERIAL PRIMARY KEY,
"biletID" INTEGER,
"odemeTarihi" DATE,
"tutar" DECIMAL,
CONSTRAINT "fkBiletID" FOREIGN KEY("biletID") REFERENCES "Biletler"("bi
letID")
ON DELETE CASCADE ON UPDATE CASCADE
);
```

▼ Fonksiyonlar | 2 Fonksiyon

1.surucuPuanOrtalamasiHesapla

```
CREATE OR REPLACE FUNCTION "surucuPuanOrtalamasiHesapla"("surucuId" INT
EGER)
RETURNS DECIMAL AS $$
DECLARE
"ortalama" DECIMAL;
BEGIN
SELECT AVG("puan") INTO "ortalama"
FROM "SurucuPuani"
WHERE "surucuID"="surucuId";
RETURN "ortalama";
END;
$$ LANGUAGE plpgsql;
```

2.yeniSeferEkle

```
CREATE OR REPLACE FUNCTION "yeniSeferEkle"("arac_id" INTEGER,"surucu_i
d" INTEGER,"guzergah" VARCHAR, "tarih" DATE,"saat" TIME)
RETURNS INTEGER AS $$
DECLARE
"yeniSeferID" INTEGER;
BEGIN
INSERT INTO "Seferler"("aracID","surucuID","guzergah","tarih","saat" )
VALUES("arac_id","surucu_id","guzergah","tarih","saat")
RETURNING "seferID" INTO "yeniSeferID";
RETURN "yeniSeferID";
END;
$$ LANGUAGE plpgsql;
```

▼ Saklı Yordamlar | 2 Saklı Yordam

1.puanVer

```
CREATE OR REPLACE PROCEDURE "puanVer"("surucu_id" INTEGER, "puan" INTEG
ER)
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO "SurucuPuani"("surucuID", "puan", "tarih")
    VALUES ("surucu_id", "puan", CURRENT_DATE);
END;
$$;
```

2.bakimYap

```
CREATE OR REPLACE PROCEDURE "bakimYap"("arac_id" INTEGER, "detaylar" TE
XT)
LANGUAGE plpgsql AS $$
BEGIN
    INSERT INTO "AracBakim"("aracID", "bakimTarihi", "detaylar")
    VALUES ("arac_id", CURRENT_DATE, "detaylar");
END;
$$;
```

▼ Triggerlar | 4 Trigger

1.sikayetTarihiGuncelle

```
CREATE OR REPLACE FUNCTION "sikayetTarihiGuncelle"()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "Sikayetler"
    SET "tarih" = CURRENT_DATE
    WHERE "sikayetID" = NEW."sikayetID";
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "sTarihiGuncelle"
AFTER INSERT ON "Sikayetler"
FOR EACH ROW
EXECUTE FUNCTION "sikayetTarihiGuncelle"();
```

2.yorumTarihiGuncelle

```
CREATE OR REPLACE FUNCTION "yorumTarihiGuncelle"()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "DurakYorumlari"
    SET "tarih" = CURRENT_DATE
    WHERE "yorumID" = NEW."yorumID";
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "yTarihiGuncelleTrigger"
AFTER INSERT ON "DurakYorumlari"
FOR EACH ROW
EXECUTE FUNCTION "yorumTarihiGuncelle"();
```

3.kapasiteAzalt

```
CREATE OR REPLACE FUNCTION "kapasiteAzalt"()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "Araclar"
    SET "kapasite" = "kapasite" - 1
    WHERE "aracID" = NEW."aracID";

    IF (SELECT "kapasite" FROM "Araclar" WHERE "aracID" = NEW."aracID")
```

```
< 0 THEN
        RAISE EXCEPTION 'Aracta yer kalmadi!';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "kAzaltTrigger"
AFTER INSERT ON "Seferler"
FOR EACH ROW
EXECUTE FUNCTION "kapasiteAzalt"();
```

4.bakimDurumuGuncelle

```
CREATE OR REPLACE FUNCTION "bakimDurumuGuncelle"()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE "Araclar"
    SET "bakimDurumu" = TRUE
    WHERE "aracID" = NEW."aracID";
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER "bakimGuncellemeTrigger"
AFTER INSERT ON "AracBakim"
FOR EACH ROW
EXECUTE FUNCTION "bakimDurumuGuncelle"();
```

*Crow s foot diagramı klasördedir