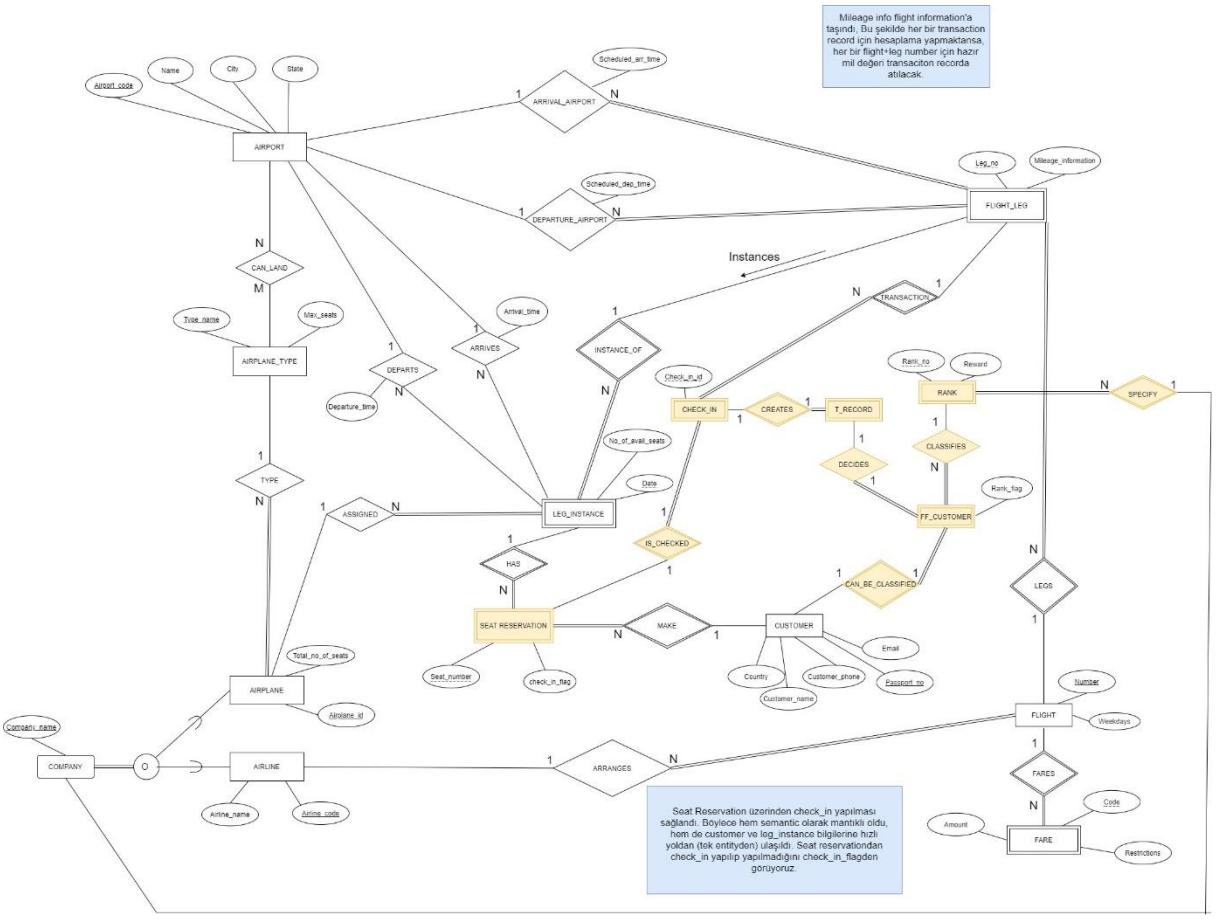
DATABASE MANAGEMENT GROUP-1 FINAL REPORT

EGE UNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ

MEHMET ANIL TAYSİ, 05170000022
TARKAN YAMAN, 05170000026
ENES ALPER BALTA, 05170000021
MUHAMMED NAIM YUSUFI, 05050007596

İçindekiler

1. EER DIAGRAM SON HALİ vc IMPLEMENTASYON



Vizedeki Diagramın revize edilmiş, SQL ortamında oluşturulmuş son hali

Link: https://github.com/MehmetAnil/Group1_Diagrams (Commit açıklamasına final olduğunu belirten not düşülmüştür)

1.1 Yapılan Değişikliklerin Açıklanması ve Customer Segmentation

Tasarım aşamasındaki EER Diagramımızın mantığı açısından büyük bir değişiklik yapılmasa da, ufak tefek pürüzler giderilmiş, SQL ortamında implementasyonu için uygun koşullar oluşturulmuştur. İmplementasyon yukarıdaki revize edilmiş Diagrama göre yapılmıştır. Vizedeki haline göre, yapılan değişiklikleri açıklayacak olursak,

CUSTOMER ve LEG_INSTANCE arasındaki HAS_TICKET ilişkisi diagramdan çıkarılmıştır. CHECK_IN varlığı artık IS_CHECKED ilişkisi ile doğrudan SEAT_RESERVATION'a bağlıdır. Böylece,

- Herhangi bir müşterinin eğer koltuk rezervasyonu var ise, SEAT_RESERVATION varlığına tanımlı check_in_flag özelliği sayesinde, eğer check-in yapmış ise 1, check-in yapmamış ise 0 gibi boolean olarak düşünülebilecek veriler tutulup, müşterinin CHECK_IN varlığına aktarılıp aktarılmayacağı konusunda bir karar mekanizması yeteneğini veritabanımıza kazandırmış oluyoruz.

- Eğer müşteri check-in yapmış, fakat iptal etmek istiyorsa, SEAT_RESERVATION üzerinden ilgili kişinin ilgili flight ve leg number'ına göre check_in_flag özelliğini 0'a eşitlemek suretiyle o kişiyi yine ilgili flight ve leg nubmer'larından çıkarabilmeliyiz. CHECK_IN varlığı T_RECORD'u da etkileyeceğinden T_RECORD tablosuna aktarılan verilerin de bir önceki haline gelmesi gerekmektedir. Bunu sağlamak için veritabanını gerçekleştirdiğimiz ortamda TRIGGER sorguları hazırladık, sonraki bölümde bu TRIGGER'ların işlevleri açıklanacaktır.
- Bir başka değişiklik olarak, EER Diagramımıza RANK varlığını eklemek oldu. Bu RANK varlığı, COMPANY varlığından company_name'i ve kendi üzerine tanımlı partial key olan Rank_no ikilisini birleşimi şeklinde (composite PK) primary key'e sahiptir.

Company	Rank_no	Reward	Minimum_mileage
Oneworld	1	%10 Discount on cafe service	10000
Oneworld	2	An airplane model	12500
Oneworld	3	%15 pay-back	15000
Star Alliance	1	%10 Discount on cafe service	11000
Star Alliance	2	An airplane model	14500
Star Alliance	3	%15 pay-back	17500
Supreme Corp.	1	A key-holder	8500
Supreme Corp.	2	%35 discount for children	13000
Supreme Corp.	3	%15 pay-back	16500
NULL	NULL	NULL	NULL

*SELECT * FROM rank;*

Bu varlığın tasarlanma ve projeye eklenme nedeni, Her FF_CUSTOMER verisinin, düzenli uçuş yaptığı şirket farklı olabilir, bundan dolayı şirketlerin sahip olduğu hava yollarının uçuşlarına göre sınıflandırma yapılır. Örneğin, Turkish Airlines ve Air Canada, tabloda da görülebilen Star Alliance şirketinin birer havayolu olsun. Bir müşteri hem Air Canadada, hem Turkish Airlines üzerinde uçuş yaparsa, Rank kazanmak için gereken Minimum Mileage'a o kadar yaklaşır, veya uçuşun miline göre Star Alliance şirketi için RANK sahibi olur. Fakat bu aynı müşterinin Oneworld şirketi için bir düzenli müşteri olduğunu göstermez. Bu nedenle, tasarımımızda RANK varlığının olması mecburdur.

- Bir diğer önemli farklılık, FF_CUSTOMER varlığı artık CUSTOMER varlığının bir türevi olarak tanımlı değildir. Bunun yerine, FF_CUSTOMER'ı CUSTOMER varlığına bağlı bir 'Weak Entity' olarak düşünüp, 'can_be_classified' gibi bir ilişki ile iki varlığı bağlamış olduk. Bunun sebebi ise, FF_CUSTOMER'ı eğer CUSTOMER üzerinden türetirsek, ona bağlı tüm bilgileri/attributeları çekmek zorunda kalmamızdır. CUSTOMER varlığının çok fazla attribute'u vardır ve bunların hepsinin FF_CUSTOMER'a aktarılmasına gerek yoktur. FF_CUSTOMER varlığının tablo gösterimine bakacak olursak.

Cust_pass	Company	Rank_no
90899545	Oneworld	2
NULL	NULL	NULL

*SELECT * FROM ff_customer;*

Cust_pass FK'i üzerinden CUSTOMER tablosunda iligli müşteriye ait özelliklere bakılabilir.

Yapılan son değişiklikler ile, TRIGGER'lar yardımı ile gerçekleştireceğimiz 'Customer Segmentation' işlemini COMPANY varlıklarına da bağlı olacak şekilde ayarladığımızdan, daha akla uygun ve gerçekçi bir sistem geliştirmiş olduğumuzu düşünüyoruz.

1.2 T_RECORD Varlığının Mantığı

Check_in_id	Customer_pass	Flight_number	Leg_number	Date	Seat_number	Mileage
29	90899545	6568	1	2020-08-22	112	2095.3
30	90899545	6568	2	2020-08-22	10	1000
37	90899545	4543	1	2020-07-08	56	355
48	90899545	4543	2	2020-07-09	53	5798
NULL	NULL	NULL	NULL	NULL	NULL	NULL

*SELECT * FROM check_in;*

T_RECORD'u anlamak için öncelikle CHECK_IN tablosuna bakalım. Check_in_id adında bir 'partial key'e sahiptir. Kalan tüm özelliklerini (Mileage dışında) SEAT_RESERVATION'dan composite foreign key olarak alır, sonuçta bu varlığın oluşmasını sağlayan varlık odur. Burada ek olarak, Check-in işlemi yapılmış bir müşteriye ait her uçuş için mil bilgisi yer alır, bu bilgiyi mevcut Flight_number + Leg_number (Flight_leg'deki PK) değerleri ile ilgili yerden alır.

Customer_pass	Company	Mileage_per_company
90899545	Oneworld	6153
90899545	Supreme Corp.	3095.3
NULL	NULL	NULL

*SELECT * FROM t_record;*

T_RECORD tablosu Customer_pass, Company ve Mileage_per_company verilerini bünyesinde barındırır. Çalışma mantığı tamamen TRIGGER'lara bağlıdır. Eğer CHECK_IN tablosuna bir varlık eklenirse;

- Öncelikle o varlık T_RECORD tablosunda var mı diye kontrol edilir, eğer yok ise eklenir. Buradan şu sonuç çıkabilir, aynı Customer_pass ve Company değerleri için sadece bir kez T_RECORD oluşturulur.
- Eğer Tabloda CHECK_IN yapmış olan Customer verisi ve ona bağlı olarak uçuşun yapıldığı Flight_number'a göre Company verisi varsa, CHECK_IN'deki Mileage değerine göre Mileage_per_company Sütununda kümülatif olarak toplanır.

Böylece, Müşterilerin firmalara bağlı olarak toplam millerini tek tabloda tutmuş oluyoruz. Burada hareketle, FF_CUSTOMER belirlenmesi işlemi, RANK tablosundaki her bir Company'e bağlı Rank değerleri, eğer T_RECORD tablosunda herhangi bir müşterinin ilgili firmayla ilişkili minimum_mileage değerini geçerse TRIGGER tetiklenir ve bir adet FF_CUSTOMER varlığı oluşur. Eğer müşteri aynı firmayla uçmaya devam ederse, Rank_no Artışı da gözlemlenebilmektedir.

Ayrıca, 2. Bölümde verilecek TRIGGERLARA bağlı değişkenler, yani üzerine INSERT, DELETE, UPDATE eventları gerçekleştirilecek tüm veriler CASCADE (eğer strong entity'deki veri değişime uğrarsa, weak entity'deki de değişir) şeklinde tanımlanmıştır.

2. TRIGGERLARIN KULLANIMI

Veritabanı gerçekleştirimi esnasında kullanılan tüm komutlar .sql dosyaları halinde proje klasöründe bulunabilir. Ayrıca tabloların oluşturulduğu ve ilişkilerin tanımladığı **airlinedb** database dosyası programdan export edilmiş bir halde teslim edilecektir.

Bu bölümde Customer Segmentation için tanımlanan 6 adet Triggera yer verilmiştir.

```
/* İlk triggerda hedef seat_reservation tablosundaki verileri check_in tablosuna 'check_in_flag' degerine gore aktarma iselmi yapmak. */

DELIMITER $$
CREATE TRIGGER after_seatreservation_insert AFTER INSERT ON seat_reservation
FOR EACH ROW
BEGIN
    DECLARE millpoint FLOAT;

    SELECT Mileage_information
    INTO millpoint
    FROM flight_leg
    WHERE flight_leg.Flight_number=NEW.Flight_number AND
    flight_leg.Leg_number=NEW.Leg_number;

    SET SQL_SAFE_UPDATES=0;
    IF NEW.check_in_flag=1 THEN
        INSERT INTO check_in SET Customer_pass = NEW.Customer_pass, Flight_number = NEW.Flight_number, Leg_number = NEW.Leg_number,
        Date = NEW.Date, Seat_number= NEW.Seat_number, Mileage = millpoint;
    END IF;
```

AFTER INSERT ON seat_reservation

SEAT_RESERVATION varlığına bir ekleme yapıldığı takdirde, eğer check_in_flag=1 ise, bu eklenen veri CHECK_IN tablosuna ilgili Flight_number ve Leg_number'a göre mileage information ile eklenir.

```
DELIMITER $$
CREATE TRIGGER after_seatreservation_update AFTER UPDATE ON seat_reservation
FOR EACH ROW
BEGIN
    DECLARE millpoint FLOAT;

    SELECT Mileage_information
    INTO millpoint
    FROM flight_leg
    WHERE flight_leg.Flight_number=NEW.Flight_number AND
    flight_leg.Leg_number=NEW.Leg_number;

    SET SQL_SAFE_UPDATES=0;
    IF NEW.check_in_flag=1 THEN
        INSERT INTO check_in SET Customer_pass = NEW.Customer_pass, Flight_number = NEW.Flight_number, Leg_number = NEW.Leg_number,
        Date = NEW.Date, Seat_number= NEW.Seat_number, Mileage = millpoint;
    ELSEIF NEW.check_in_flag=0 THEN
        DELETE FROM check_in WHERE Customer_pass = NEW.Customer_pass AND Flight_number = NEW.Flight_number AND Leg_number = NEW.Leg_number;
    END IF;
 SET SQL_SAFE_UPDATES=1;
END$$
DELIMITER ;
```

AFTER UPDATE ON seat_reservation

Eğer müşteri check ini iptal ederse, yani check_in_flag = 0 olacak şekilde bir UPDATE işlemi söz konusu olursa, ilgili CHECK_IN verisi tablodan silinir. Tersi durumda, yani var olan bir seat_reservation verisi check_in_flag = 1 olacak şekilde UPDATE edilirse, ilgili veri Check_in tablosuna aktarılır.

```
/* check_in insert updatten sonra */
DELIMITER $$
CREATE TRIGGER after_checkin_insert AFTER INSERT ON check_in
FOR EACH ROW
BEGIN
    DECLARE comp_name VARCHAR(255);
    SET SQL_SAFE_UPDATES=0;
    SELECT Company
    INTO comp_name
    FROM flight
    WHERE flight.Flight_number = NEW.Flight_number;

    IF (SELECT EXISTS(SELECT * FROM T_record WHERE Customer_pass = NEW.Customer_pass AND Company = comp_name) =0) THEN
        INSERT INTO t_record SET Customer_pass = NEW.Customer_pass, Company = comp_name, Mileage_per_company = NEW.Mileage;
    ELSE
        UPDATE t_record
        SET Mileage_per_company = Mileage_per_company + NEW.Mileage
        WHERE Customer_pass = NEW.Customer_pass AND t_record.Company = comp_name;
    END IF;
    SET SQL_SAFE_UPDATES=1;
END$$
DELIMITER ;
```

AFTER INSERT ON check_in

Eğer check_in'e yazılan bir veri, T_Record tablosunda bulunmuyorsa (SELECT EXISTS kullanılarak yapılmıştır) bu veri ilgili customer_passport ve company değerleri bulunarak T_RECORD tablosuna yazılır. Eğer bu tabloda zaten belirtilen şekilde bir veri varsa, yeni yapılan check_in işleminin Mileage değeri mevcut değerle toplanıp aynı yere yazılır, böylece toplam mil değeri takibi sağlanmış olur.

```
DELIMITER $$
CREATE TRIGGER after_checkin_delete AFTER DELETE ON check_in
FOR EACH ROW
BEGIN
    DECLARE comp_name VARCHAR(255);
    SET SQL_SAFE_UPDATES=0;
    SELECT Company
    INTO comp_name
    FROM flight
    WHERE flight.Flight_number = OLD.Flight_number;

    IF (SELECT EXISTS(SELECT * FROM T_record WHERE Customer_pass = OLD.Customer_pass AND Company = comp_name) =1) THEN
        UPDATE t_record
        SET Mileage_per_company = Mileage_per_company - OLD.Mileage
        WHERE Customer_pass = OLD.Customer_pass AND t_record.Company = comp_name;
        IF (SELECT EXISTS(SELECT Mileage_per_company FROM t_record WHERE Customer_pass = OLD.Customer_pass AND Company = comp_name) = 0) THEN
            DELETE FROM t_record WHERE Customer_pass = OLD.Customer_pass AND Company = comp_name;
        END IF;
    END IF;
    SET SQL_SAFE_UPDATES=1;
END$$
DELIMITER ;
```

AFTER DELETE ON check_in

Yukarıda check_in verilerinin silinebileceğinden bahsetmiştik. Eğer, bir Check_in verisi silinirse, Eğer Mileage_per_company değeri silinen Check_in mil değeri ile aynıysa, veri T_RECORD'dan da silinir. Değilse, silinen check_in'in mil değeri T_RECORD tablosundaki mil değerinden çıkarılır ve tablo update edilir.

```
/* t_record tablosunu ilgilendiren udpate ve insert triggerları*/
DELIMITER $$
CREATE TRIGGER after_trecord_insert AFTER INSERT ON t_record
FOR EACH ROW
BEGIN
    DECLARE max_rank INT;
    SET SQL_SAFE_UPDATES=0;

    SELECT MAX(Rank_no)
    INTO max_rank
    FROM airlinedb.rank
    WHERE Company = NEW.Company AND NEW.Mileage_per_company > Minimum_mileage;

    IF(max_rank > 0) THEN
        IF(SELECT EXISTS(SELECT Cust_pass,Company FROM ff_customer WHERE Cust_pass = NEW.Customer_pass AND Company = NEW.Company) = 0) THEN
            INSERT INTO ff_customer SET Cust_pass = NEW.Customer_pass, Company = NEW.Company, Rank_no = max_rank;
        ELSE
            UPDATE ff_customer SET Cust_pass = NEW.Customer_pass, Company = NEW.Company, Rank_no = max_rank;
        END IF;
    END IF;

    SET SQL_SAFE_UPDATES=1;
END$$
DELIMITER ;
```

AFTER INSERT ON t_record

T_Recorda eklenen verinin mileage_per_company değeri eğer RANK tablosunda bağlı olduğu company'e göre ayarlanmış rank değerlerinin minimum_mileage koşulunu sağlıyor ise, bu veri eşleştiği rank'e göre FF_CUSTOMER tablosuna yazılır. Aslında 'Customer Segmentation' mantığının son kısmı da budur. Bu tabloya bağlı bir de UPDATE trigger'ı vardır, fakat mantık olarak aynı şeyi yapar. Örneğin Check_in silindiği durumda minimum_mileage durumu sağlanmaz hale gelirse o customer FFC'den silinir. AFTER UPDATE trigger'ı aşağıda verilmiştir.

```
CREATE TRIGGER after_trecord_update AFTER UPDATE ON t_record
FOR EACH ROW
BEGIN
    DECLARE max_rank INT;
    DECLARE min_rank INT;
    SET SQL_SAFE_UPDATES=0;
    SELECT MAX(Rank_no)
    INTO max_rank
    FROM airlinedb.rank
    WHERE Company = NEW.Company AND NEW.Mileage_per_company > Minimum_mileage;
    SELECT MIN(Rank_no)                          /* min_rank Check_in varligi silinmesi dolayisiyla ff_customerdan cikarilmasi gereken varlik icin tani
    INTO min_rank
    FROM airlinedb.rank
    WHERE Company = NEW.Company;
    IF(NEW.Mileage_per_company < (SELECT Minimum_mileage FROM airlinedb.rank WHERE Company = NEW.Company AND Rank_no = min_rank)) THEN
            DELETE FROM ff_customer WHERE Cust_pass = NEW.Customer_pass AND Company = NEW.Company;
    ELSE
        INSERT INTO ff_customer SET Cust_pass = NEW.Customer_pass, Company = NEW.Company, Rank_no = max_rank ON DUPLICATE KEY UPDATE Rank_no = max_rank;
    END IF;
    SET SQL_SAFE_UPDATES=1;
END$$
```

AFTER UPDATE ON t_record

Update trigger'i insert'te olduğu gibi yine ff_customer'ın durumunu tamamiyle belirler. Eğer değişen t_record mileage değeri aynı company için minimum rank değerinden daha küçük ise (artık) veri ff_customer'dan silinir.

```
#T_RECORD SİLİNİRSE FF_CUSTOMER'DA SİLİNMELİ
DELIMITER $$
CREATE TRIGGER after_trecord_delete AFTER DELETE ON t_record
FOR EACH ROW
BEGIN
    DELETE FROM ff_customer WHERE Cust_pass = OLD.Customer_pass AND Company = OLD.Company;

END$$
DELIMITER ;
```

AFTER DELETE ON t_record

Eğer t_record tablosundaki bir müşteriye ait t_record verisi silinecek olursa, ki eğer o müşteri için check_in varlığı iptal edilirse silinmesi gerekiyor, aynı veriler için eğer daha önceden oluşmuşsa ff_customer varlığı da siliniyor.

```
#Uygun koltuk sayısından bir tane düşüldü
UPDATE leg_instance
SET Number_of_available_seats = Number_of_available_seats - 1
WHERE Flight_number = NEW.Flight_number AND Leg_number = NEW.Leg_number;
```

AFTER INSERT check_in (EKLEME)

Ayrıca artık yukarıdaki şekilde de gördüğünüz gibi, check_in tablosuna veri geldiğinde, check-in yapılan uçuş için, leg_instance'da bulunan number of available seats özelliği bir azalacak şekilde tasarlandı, böylece her leg instance için belirli sayıda müşteri gelebiliyor olacak.

```
#Uygun koltuk sayısından bir tane arttırıldı
UPDATE leg_instance
SET Number_of_available_seats = Number_of_available_seats + 1
WHERE Flight_number = OLD.Flight_number AND Leg_number = OLD.Leg_number;
```

AFTER DELETE check_in (EKLEME)

Eğer check-in iptal edilirse ve check_in tablosundaki veri silinirse, silinen veri için geçerli koltuk sayısı eski haline döndürülmek için +1 ekleme yapıldı.

3. CHECK CONSTRAINTS

Check Constraints Komutları aşağıdaki gibidir.

```
/* Herhangi bir sivil uçağın koltuk sayısı maksimum 800 olabilir */
ALTER TABLE airplane
ADD CONSTRAINT CHK_seat_num CHECK (Total_number_of__seats < 800);

/* Boş koltuk sayısı sıfırın altında olamaz */
ALTER TABLE leg_instance
ADD CONSTRAINT CHK_available_seat CHECK (Number_of_available_seats  >= 0);
//Seat numberdan check-in yapıldıkça düşüldüğü için 0'ın altına inmemesi sağlandı.

/*Customer için Country Code 3 haneli olmalıdır */
ALTER TABLE customer
ADD CONSTRAINT CHK_countryCode CHECK (char_length(Country_code) = 3);

/* Ucret sıfırdan kucuk, 7000'den buyuk olamaz. 7000 tavan fiyat seçildi */
ALTER TABLE fare
ADD CONSTRAINT CHK_fare CHECK ( Amount < 7000 AND Amount> 0);

/* Dünya üzerindeki en kısa ve en uzun tek seferlik uçuş baz alınarak yapılmıştır */
ALTER TABLE flight_leg
ADD CONSTRAINT CHK_mileage_info CHECK ( Mileage_information > 1.7 AND
Mileage_information < 9.534);
```

Koşul kısıtlamaları sadece tanımlandığı tablodaki veriler ile kullanılabildiğinden, genel olarak o tablodaki verileri belirli bir sınırı aşamayacak veya belirli bir sınırdan altına inemeyecek şekilde tanımlamışlardır.

4. INSERT UPDATE DELETE

```
/* INSERT STATEMENTS FOR 5 TABLES */
INSERT INTO airlinedb.airport (Airport_code, Name, City, State)
VALUES ('JFK', 'John F. Kennedy International', 'New York', 'USA');

INSERT INTO airlinedb.airline (Airline_code, Company, Airline_name)
VALUES ('252', 'Taysi A.S', 'Nallihan Airlines');

INSERT INTO airlinedb.flight (Flight_number, Airline_code, Company, Weekdays)
VALUES ('1000', '252', 'Taysi A.S', 'Monday,Saturday,Sunday');

INSERT INTO airlinedb.flight_leg (Flight_number, Leg_number, Departure_airport_code, Scheduled_depature_time,
Arrival_airport_code, Scheduled_arrival_time, Mileage_information)
VALUES ('1000', '1', 'JFK', '13:15:00', 'ISL', '01:30:00', '5005');

INSERT INTO airlinedb.fare (Flight_number, Fare_code, Amount, Restrictions)
VALUES ('1000', 'CH', '50', 'Child Fare'),('1000', 'F', '235', 'First Class'); /* Multiple Rows */
```

Insert Events

```
/* UPDATE STATMENTS FOR 5 TABLES */
UPDATE airlinedb.fare SET Amount = '45'
WHERE (Flight_number = '4543') and (Fare_code = 'W');

UPDATE airlinedb.airplane SET Total_number_of__seats = '135'
WHERE (Airplane_id = '113') and (Company = 'Airbus');

UPDATE airlinedb.rank SET Reward = '%20 Grocery discount for ABC Market'
WHERE (Company = 'Supreme Corp.') and (Rank_no = '3');

UPDATE airlinedb.flight_leg SET Scheduled_depature_time = '18:15:00'
WHERE (Flight_number = '6569') and (Leg_number = '1');

UPDATE airlinedb.check_in
SET Check_in_flag = '0';        /*Check_in_flag attribute set to 0 for all Columns. */
```

Update Events

```
/* DELETE STATEMENTS FOR 5 TABLES */
DELETE FROM airlinedb.customer
WHERE Passport_no = '14867898';

DELETE FROM airlinedb.airplane
WHERE Airplane_id = '55';

DELETE FROM airlinedb.seat_reservation
WHERE (Customer_pass = '90147895') and (Leg_number = '1') and (Flight_number = '4543') and (Date = '2020-07-08') and (Seat_number = '45');

DELETE FROM airlinedb.fare
Where Fare_code = 'CH' and Fare_code = 'F'
ORDER BY Amount
LIMIT 3;                    /*Fare Code CH and F selected, then Ordered by Amount Descedingly, Deleted first 3 rows. */

DELETE FROM airlinedb.company;      /* This query will delete all existing Company rows */
```

Delete Events

5. SELECT KOMUTLARI ve VIEW

5.b.i

/*1. Şu anda database'e kayıtlı uçuşların verilerini alır ve fiyatı artan olarak sıralar */

```
SELECT fare.Flight_number,Airline_code,Fare_code,Amount
FROM airlinedb.flight,airlinedb.fare
WHERE fare.Flight_number=flight.Flight_number
ORDER BY Amount ASC;
```

/*2. Database'e kayıtlı uçakların tiplerine göre kaçar tane olduğunu belirler */

```
SELECT Airplane_type_name,COUNT(airplane.Airplane_type) as in_airplane
FROM airplane,airplane_type
WHERE airplane.Airplane_type = airplane_type.Airplane_type_name
GROUP BY Airplane_type_name;
```

/*3. Tüm tiplerde uçak inebilen havaalanlarının kodları ve isimleri nedir? */

```
SELECT airport.airport_code, name
FROM airport, can_land
WHERE airport.airport_code = can_land.airport_code
GROUP BY airport.airport_code
HAVING COUNT(*) = 3;
```

/*4. FRA - MAN Rotasında fare kodu W(Premium economy) uçuşların belirlenmesi */

```
SELECT flight1.Flight_number, flight1.Departure_airport_code,
flight3.Arrival_airport_code, COUNT(*) AS count_leg, fare.Amount
FROM flight_leg flight1
INNER JOIN flight_leg flight2
   ON flight1.flight_number = flight2.flight_number
   AND flight2.departure_airport_code = "FRA"
INNER JOIN flight_leg flight3
   ON flight1.flight_number = flight3.flight_number
   AND flight3.arrival_airport_code = "MAN"
INNER JOIN fare
   ON flight1.flight_number = fare.flight_number
   AND fare.fare_code = "W"
GROUP BY flight_number;
```

/*5. FRA havalanının kalkış noktası olduğu, First Class tipinde uçuşları ve fiyatını belirler */

```
SELECT fare.Flight_number,Depature_airport_code,Arrival_airport_code,Date,Amount
FROM flight, fare,leg_instance
WHERE flight.Flight_number = fare.Flight_number  AND Fare_code= 'F' AND
flight.Flight_number = leg_instance.Flight_number
AND Depature_airport_code = 'FRA';
```

/*6. Belli tarihler arasında uçuç yapmak isteyen birisi için fiyat ve restrictions tablosu*/

```
SELECT leg_instance.Flight_number,leg_instance.Leg_number,(Restrictions),Amount
FROM airlinedb.fare
inner join flight on fare.Flight_number=flight.Flight_number
inner join flight_leg on flight.Flight_number=flight_leg.Flight_number
inner join leg_instance on flight_leg.Flight_number=leg_instance.Flight_number and
flight_leg.Leg_number=leg_instance.Leg_number
WHERE (leg_instance.Date BETWEEN '2020-07-07' AND '2020-07-08');
```

/*7. arrival airport için şirketler*/

```
select airport.Name,airline.company from airport
inner join flight_leg on airport.Airport_code=flight_leg.Arrival_airport_code
inner join flight on flight_leg.Flight_number=flight.Flight_number
inner join airline on flight.Airline_code=airline.Airline_code
and flight.Company=airline.Company;
```

/*8 Ülkesine dönen müşterilerin listesi */

```
SELECT customer.Country, customer.Name, customer.Phone,
flight_leg.Arrival_airport_code
FROM customer, seat_reservation, flight_leg, airport
WHERE customer.Passport_no = seat_reservation.Customer_pass
AND seat_reservation.Flight_number = flight_leg.Flight_number
AND seat_reservation.Leg_number = flight_leg.Leg_number
AND flight_leg.Arrival_airport_code = airport.Airport_code
AND customer.Country = airport.State;
```

/*9 En ucuz tarifeyle gidebileceğiniz şehirler */

```
select distinct(airport.City),flight_leg.Flight_number from airport
inner join flight_leg on airport.Airport_code=flight_leg.Arrival_airport_code
inner join flight on flight_leg.Flight_number=flight.Flight_number
inner join fare on flight.Flight_number=fare.Flight_number
where fare.Flight_number=(select fare.Flight_number from fare
group by Amount
order by Amount
limit 1);
```

/*10 */

```
SELECT company.company_name, airline.Airline_name,
flight.Flight_number,flight_leg.Leg_number,flight_leg.Mileage_information
from company,airline,flight,flight_leg
where company.company_name = airline.Company and airline.Airline_code =
flight.Airline_code and flight.Flight_number = flight_leg.Flight_number;
```

NESTED QUERIES

/*1. Bir müşterinin FFC olup olmadığını belirler, İsim bilgileri ve reward'ı yazdırır. */

```
SELECT customer.Name, Country, ff_customer.Company, ff_customer.Rank_no,
rnk.Reward

FROM customer, ff_customer, airlinedb.rank AS rnk

WHERE rnk.Rank_no = ff_customer.Rank_no AND ff_customer.Company = rnk.Company
AND (Passport_no) IN (SELECT Cust_pass

                                        FROM ff_customer);
```

/*2. Cuma ve Cumartesi uçuşu olan firmalar, uçuş kodlarının belirlenmesi, ücretleri */

```
SELECT fare.Flight_number, Amount, Restrictions

FROM fare

WHERE (Flight_number) IN (SELECT Flight_number

                                FROM flight

                                WHERE Weekdays LIKE '%Friday%' AND Weekdays LIKE
'%Saturday%');
```

/*3. First Class uçuşların ortalamalarından daha büyük ücrete sahip uçuşlar listelenir */

```
SELECT Flight_number, Amount, Restrictions

FROM fare

HAVING Amount >= all (SELECT AVG(Amount)

                                FROM fare

                                WHERE Fare_code = "F");
```

/*4. En fazla Check-in yapmış 3 müşteriyi listeler */

```
SELECT *, MAX(total_check_in)

FROM (SELECT Customer_pass, COUNT(*) as total_check_in

FROM check_in

GROUP BY Customer_pass) as total_check_in_per_customer, customer;
```

/*LEFT JOIN*/

```
SELECT
airplane.Airplane_id,airplane.Name,leg_instance.Flight_number,leg_instance.Leg_number,leg_instance.Date,leg_instance.Number_of_available_seats

From airplane

LEFT JOIN leg_instance ON airplane.Airplane_id= leg_instance.Airplane_id

ORDER by airplane.Airplane_id;
```

/*RIGHT JOIN*/

```
SELECT
airplane.Airplane_id,airplane.Name,can_land.Airplane_type_name,can_land.Airport_code,airport.Name,airport.City

FROM airplane

RIGHT JOIN can_land on can_land.Airplane_type_name = airplane.Airplane_type

RIGHT JOIN airport on can_land.Airport_code = airport.Airport_code;
```

/*FULL OUTER JOIN*/

/* customerların ffc'de bulunup bulunmadığını gösteren sorgu */

```
SELECT *

FROM customer

RIGHT JOIN ff_customer ON customer.Passport_no = ff_customer.Cust_pass

UNION

SELECT *

FROM customer

LEFT JOIN ff_customer ON customer.Passport_no = ff_customer.Cust_pass;
```

/*EXIST */

/* Eğer ISL'den kalkan bir uçağa kayıtlı müşteri varsa o uçuşun bilgilerini yazdırsın */

```
SELECT fl.Flight_number, Customer_pass, Departure_airport_code,Arrival_airport_code, Mileage_information

FROM flight_leg AS fl, seat_reservation AS sr

WHERE EXISTS (SELECT seat_reservation.Customer_pass as cust
```

```
        FROM seat_reservation,leg_instance
      WHERE seat_reservation.Flight_number = leg_instance.Flight_number AND
leg_instance.Depature_airport_code = "ISL")
      AND Departure_airport_code="ISL" AND fl.Flight_number = sr.Flight_number;
```

/*EXIST */

```
SELECT customer.Passport_no,customer.Name,ff_customer.Company
FROM customer,ff_customer
WHERE EXISTS(SELECT ff_customer.Cust_pass
            FROM ff_customer
            WHERE customer.Passport_no = ff_customer.Cust_pass and
            ff_customer.Rank_no > 2);
```

/*NOT EXIST */

```
SELECT company.company_name
FROM company
WHERE NOT EXISTS(SELECT airplane.Airplane_id
                  FROM airplane
                  WHERE company.company_name = airplane.Company)
```

VIEWS

/*1. Supreme Corp. fimrasının uçuşlarını listeler */

```
CREATE VIEW SupremeCorpFlights AS
SELECT f.Company,f_leg.Flight_number, Leg_number, Departure_airport_code,
Arrival_airport_code
FROM flight_leg AS f_leg
JOIN flight AS f
ON f.Flight_number = f_leg.Flight_number
WHERE f.Company = "Supreme Corp.";
```

/*2. Türkiyedeki havalimanlarını listeler */

```
CREATE VIEW AirportsTurkey AS
SELECT * FROM airport
WHERE State = "TUR";
```

/*3. Türkiye için iç hatlar uçuşlarını ve tarihlerini gösterir */

```
CREATE VIEW IcHatlarTurkiye AS
SELECT Flight_number,Leg_Number, Date, Depature_airport_code, Arrival_airport_code,
airport1.State
FROM leg_instance
INNER JOIN airport AS airport1 ON Depature_airport_code = airport1.Airport_code
INNER JOIN airport AS airport2 ON Arrival_airport_code = airport2.Airport_code
WHERE airport1.State = "TUR" AND airport2.State = "TUR";
```

/*4. Tüm tiplerde uçak inebilen havalanı kodları ve isimleri */

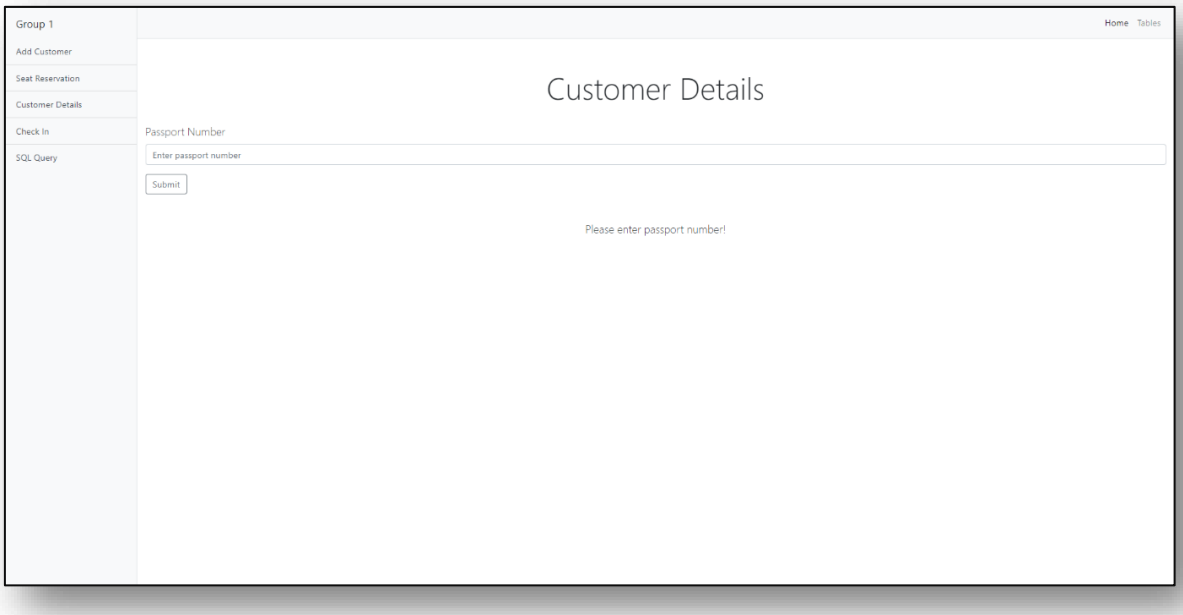
```
CREATE VIEW CokYonluHavaalanları AS
SELECT airport.airport_code, name
FROM airport, can_land
WHERE airport.airport_code = can_land.airport_code
GROUP BY airport.airport_code
HAVING COUNT(*) = 3;
```

/*5. Turk Pasaportuna Sahip Customer'ların yaptığı rezervasyonlar */

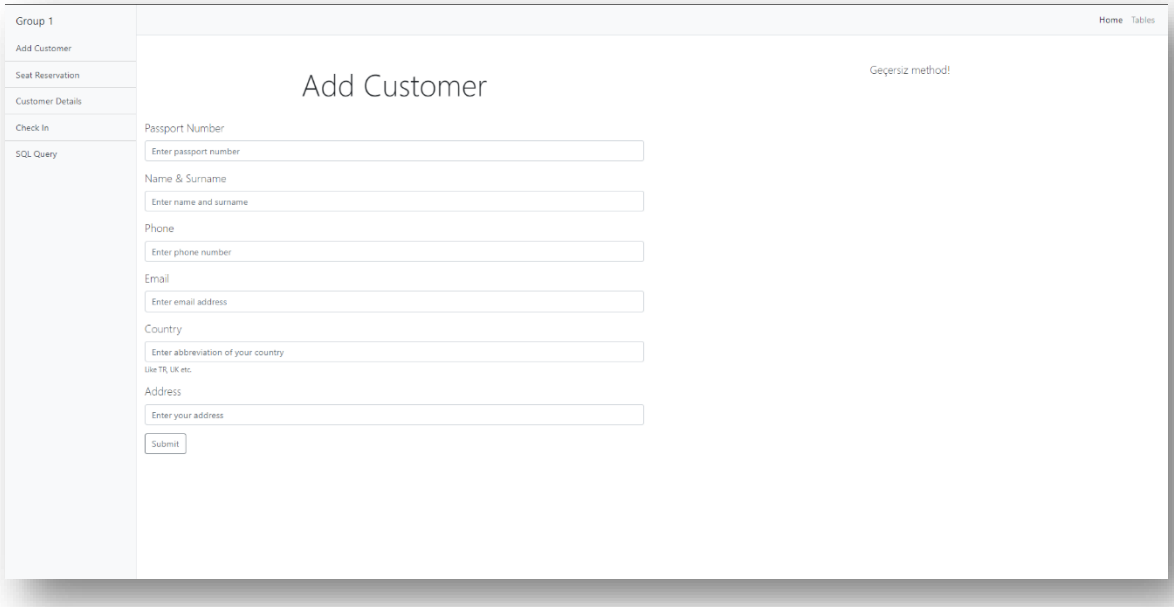
```
CREATE VIEW TurkYolcular AS
SELECT customer.Name,Flight_number, Leg_number, Date, Seat_number
FROM seat_reservation,customer
WHERE Customer_pass = customer.Passport_no AND customer.Country = "TUR";
```

6. WEB - REST API

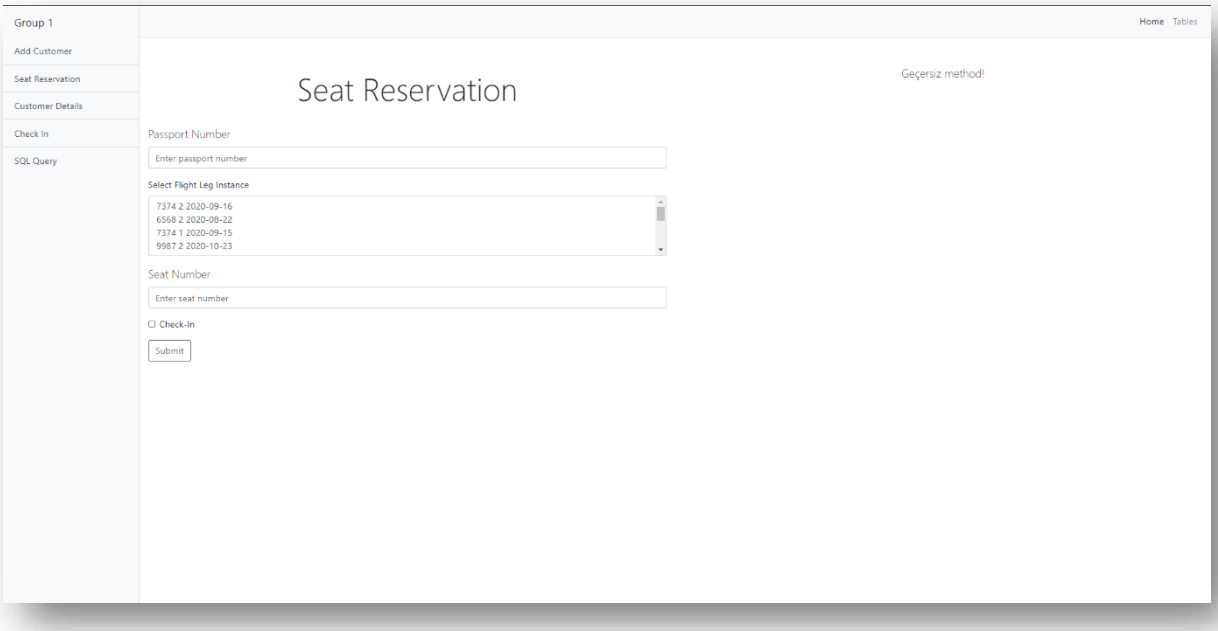
Programın web kısmında ekran görüntüleri ile arayüzün işlevlerinden bahsedilecektir.



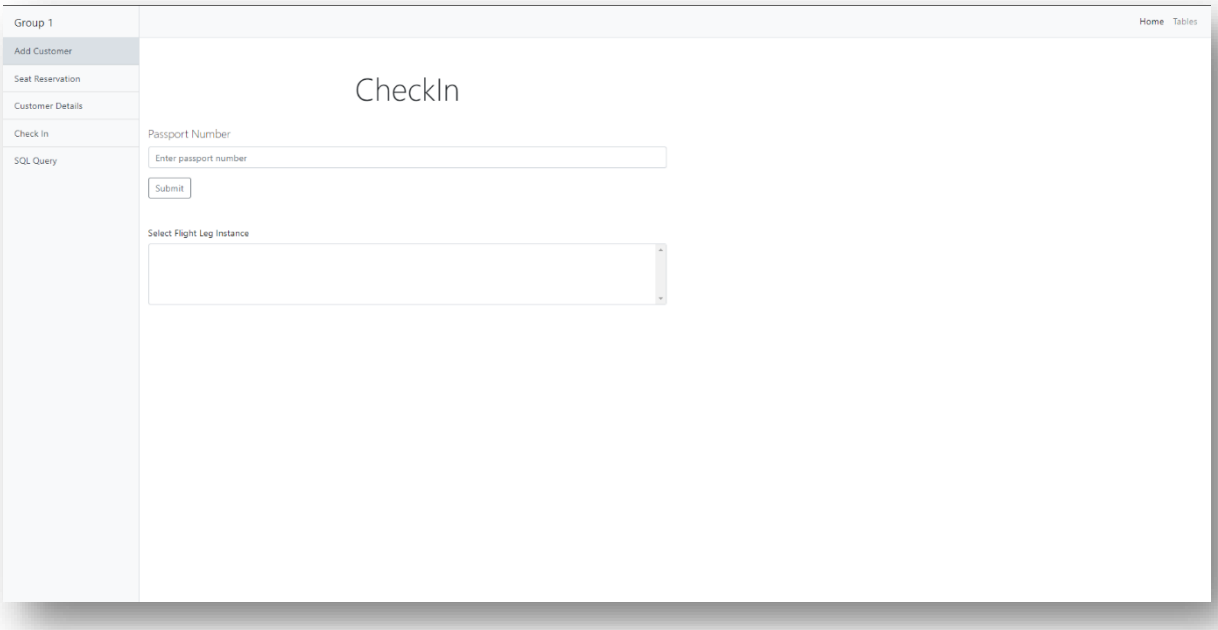
Bu sayfada, müşterinin pasaport numarasını girerek kayıtlı olan bilgilerini sorguluyoruz. Bu bilgileri bir tablo şeklinde ekrana bastırıyoruz. Eğer pasaport numarası girilen müşteri bir FFC olarak kayıtlı ise ayrıca Rank_no ve Reward bilgileri de gösterilmektedir. Ayrıca REST API üzerindeki hata veya bilgilendirme metinleri de hemen alt tarafta gösterilmektedir.



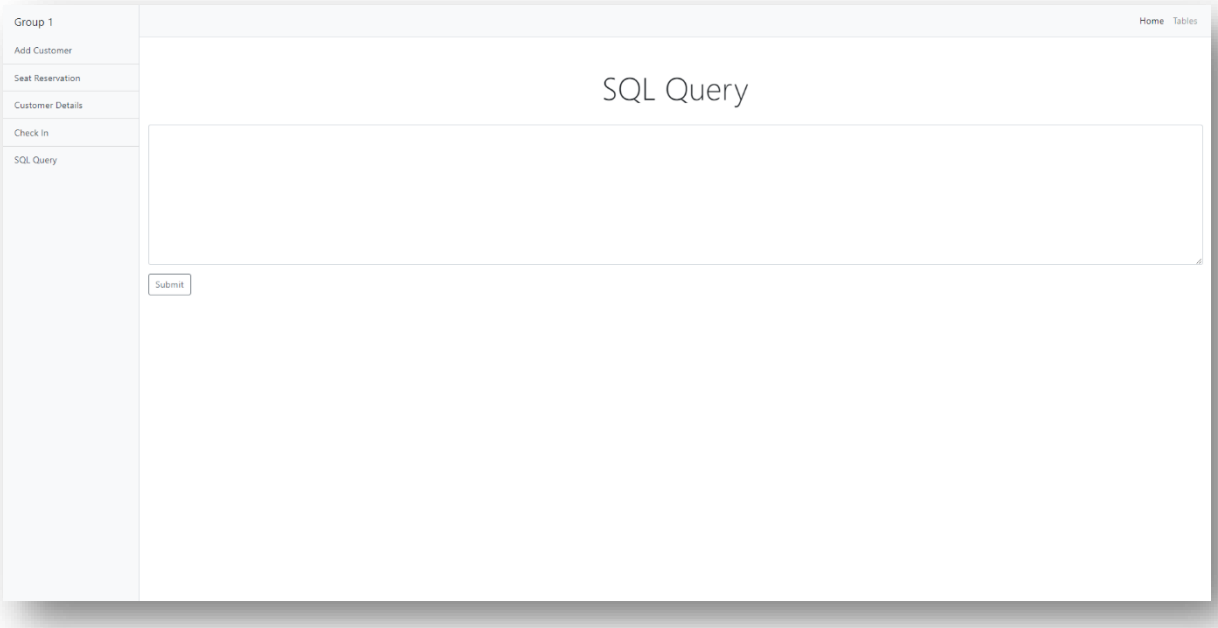
Bu sayfada, müşterimizin gerekli bilgilerini girerek veritabanımıza kaydını gerçekleştiriyoruz. REST API ile verilerimiz gönderilip veritabanına kayıt işlemi gerçekleştiriliyor. Ayrıca sol tarafta REST API üzerinden gelen bilgilendirme metni de yer almaktadır.



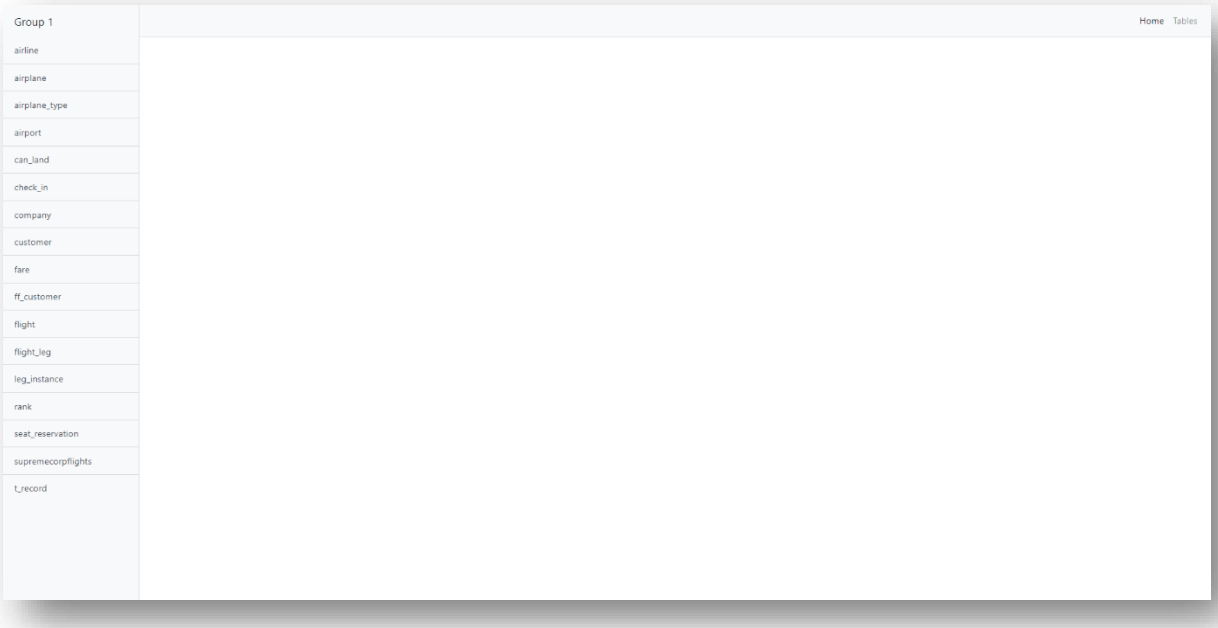
Bu sayfada, veri tabanında kayıtlı olan bir müşterimizin koltuk rezervasyonunu yapabiliriyoruz. Kullanıcı pasaport numarasını, veri tabanında kayıtlı olan uçuş ayaklarından bir tanesini, koltuk numarasını ve ayrıca check-in yapıp yapmadığını belirterek kaydı gerçekleştirmektedir.



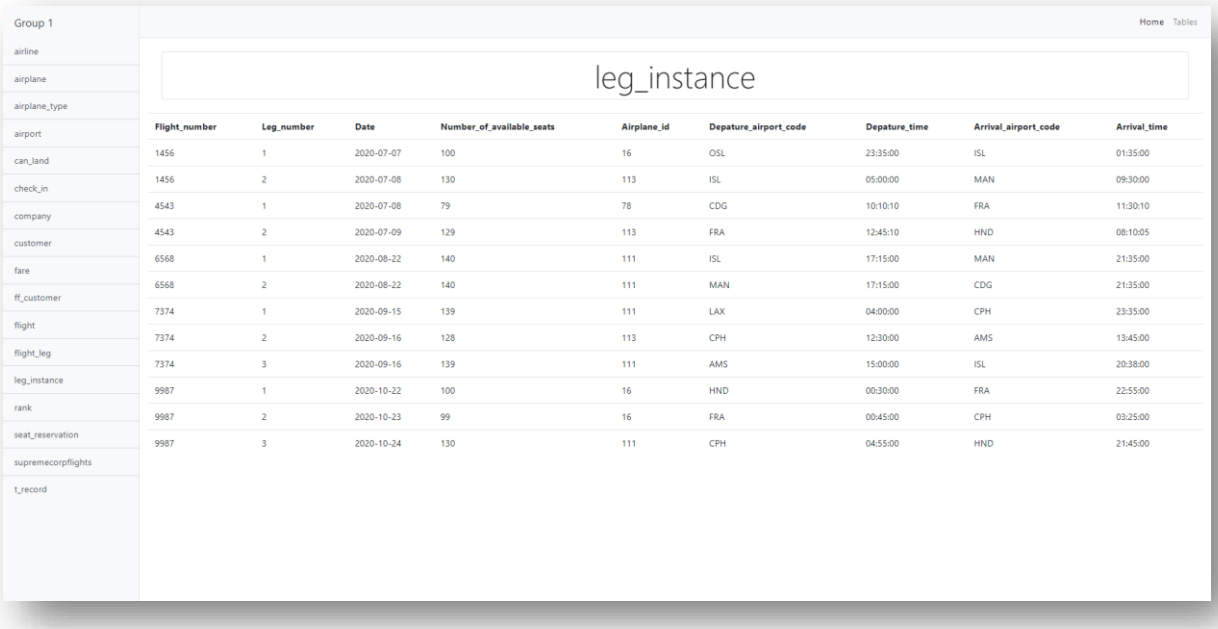
Bu sayfada, müşteri kendi pasaport numarasına kayıtlı olan uçuşlarına checkin işlemini gerçekleştirebiliyor. Ayrıca yine kendi pasaport numarasına kayıtlı olan uçuşlardaki checkin işlemini iptal edebiliyor.



Bu sayfada, veritabanı sistemimizde gerçekleştirebildiğimiz gibi SQL sorgularımızı yazabildiğimiz ve yazdığımız sorgu sonuçlarını tablo şeklinde görebildiğimiz bir arayüz bulunmaktadır.



Bu sayfada, veritabanımızdaki tüm tabloların isimleri bulunmaktadır. Bu tabloların isimlerinin üzerine tıklayarak tablo içeriklerine erişebiliriz.



Bir önceki tablo sayfamızdaki tablo isimlerine tıklandıktan sonra gelecek olan sorgu sonuçları ekranda yer almaktadır. İçerisindeki tüm satır ve sütünlar ekrana bastırılmaktadır.