CHAPTER 8 BÖLÜM SONU SORULAR

1. Bir birincil anahtar beyan edildiğinde ne tür bir bütünlük
uygulanır?
Birincil anahtar (Primary Key) tanımlandığında Varlık Bütünlüğü (Entity Integrity) sağlanır. Birincil anahtar NULL olamaz ve benzersiz olmalıdır.

2. Yalnızca rakamlar içeren bir niteliği sayısal veri türü
yerine karakter veri türü olarak bildirmenin neden daha
uygun olabileceğini açıklayın.
Yalnızca rakamlar içeren bir niteliğin karakter (CHAR/VARCHAR) olarak tanımlanması daha uygundur, çünkü:
Ön sıfırlar kaybolmaz (Örn: 00123).
Matematiksel işlem yapılmaz.
Telefon numarası, kimlik numarası gibi veriler sayısal işlem gerektirmez.

3. Sütun kısıtı ile tablo kısıtı arasındaki fark nedir?
Sütun kısıtı (Column Constraint) sadece tek bir sütuna uygulanır, Tablo kısıtı (Table Constraint) ise birden fazla sütunu kapsayabilir.

4. "Referans kısıtlama eylemleri" nedir?
Referans Kısıtlama Eylemleri (Referential Constraint Actions), foreign key ilişkileri için kullanılır:

ON DELETE CASCADE → Bağlı verileri de siler.
ON DELETE SET NULL → Bağlı verileri NULL yapar.
ON DELETE RESTRICT → Silmeyi engeller.

5. CHECK kısıtlamasının amacı nedir?
CHECK Constraint, bir sütunun belirli bir koşulu sağlamasını zorunlu kılar.

6. ALTER TABLE komutunun ne zaman gerekli
olabileceğini açıklayın.
ALTER TABLE, tabloyu değiştirmek için kullanılır:

Sütun eklemek: ALTER TABLE employees ADD COLUMN email VARCHAR(100);
Sütun silmek: ALTER TABLE employees DROP COLUMN email;

7. INSERT komutu ile UPDATE komutu arasındaki fark
nedir?
INSERT → Yeni veri ekler.
UPDATE → Mevcut veriyi günceller.

8. CREATE TABLE komutuyla bir alt sorgu kullanmak ile
INSERT komutuyla bir alt sorgu kullanmak arasındaki
fark nedir?
CREATE TABLE + Alt Sorgu → Yeni bir tablo oluşturur ve veri ekler.
INSERT + Alt Sorgu → Mevcut tabloya veri ekler.

9. Dizi nedir? Sözdizimini yazınız.
Sequence (Dizi), otomatik artan değerler üretir.

10. Tetikleyici nedir ve amacı nedir? Bir örnek veriniz.
Trigger (Tetikleyici), belirli bir işlem (INSERT, UPDATE, DELETE) gerçekleştiğinde otomatik olarak çalışır.

```
CREATE TRIGGER trg_check BEFORE INSERT ON employees
FOR EACH ROW WHEN (NEW.salary < 2000)
BEGIN
    RAISE EXCEPTION 'Maaş 2000'den düşük olamaz!';
END;
```

11. Saklı yordam nedir ve neden özellikle yararlıdır? Bir
örnek verin.

Stored Procedure (Saklı Yordam), tekrar tekrar çalıştırılabilen SQL kodlarıdır.

```sql
CREATE PROCEDURE increase_salary (emp_id INT, amount DECIMAL)
AS
BEGIN
    UPDATE employees SET salary = salary + amount WHERE id = emp_id;
END;
```

Çağırmak için:

```sql
CALL increase_salary(1, 500);
```

PROBLEMLER

1. EMP_1 adlı bir tablo için yalnızca tablo yapısını oluşturacak SQL kodunu yazın. Bu tablo EMPLOYEE tablosunun bir alt
kümesi olacaktır. Temel EMP_1 tablo yapısı aşağıdaki özetlenmiştir. Birincil anahtar olarak EMP_NUM kullanın. JOB_CODE'un JOB için FK olduğuna dikkat edin, bu nedenle referans bütünlüğünü uyguladığınızdan emin olun. Kodunuz
ayrıca EMP_LNAME ve EMP_FNAME içinde null girişleri önlemelidir.

EMP_1 Tablosunun Yapısını Oluşturma

```sql
CREATE TABLE EMP_1 (
    EMP_NUM VARCHAR(3) PRIMARY KEY,
    EMP_LNAME VARCHAR(15) NOT NULL,
    EMP_FNAME VARCHAR(15) NOT NULL,
    EMP_INITIAL CHAR(1),
    EMP_HIREDATE DATE,
    JOB_CODE VARCHAR(3),
    FOREIGN KEY (JOB_CODE) REFERENCES JOB(JOB_CODE)
);
```

2. Problem 1'de tablo yapısını oluşturduktan sonra, Şekil P8.2'de gösterilen tablo için ilk iki satırı girmek üzere SQL kodunu yazın. Her satır bir alt sorgu kullanmadan ayrı ayrı eklenmelidir. Satırları şekilde listelendikleri sırayla ekleyin

İlk İki Satırı Ekleme

```sql
INSERT INTO EMP_1 (EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_HIREDATE, JOB_CODE)
VALUES ('101', 'Smith', 'John', 'A', '2005-06-20', '500');

INSERT INTO EMP_1 (EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_HIREDATE, JOB_CODE)
VALUES ('102', 'Doe', 'Jane', 'B', '2007-03-15', '501');
```

3. Zaten var olan EMPLOYEE tablosunu kullanarak, EMPLOYEE tablosundan kalan satırları EMP_1 tablosuna eklemek
için bir alt sorgu kullanın. Alt sorgunuzun yalnızca EMP_1 tablosu için gereken sütunları ve yalnızca şekilde gösterilen
çalışanları alması gerektiğini unutmayın.

EMPLOYEE Tablosundan Kalan Satırları Alt Sorgu ile Ekleme

```sql
INSERT INTO EMP_1 (EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_HIREDATE, JOB_CODE)
SELECT EMP_NUM, EMP_LNAME, EMP_FNAME, EMP_INITIAL, EMP_HIREDATE, JOB_CODE
FROM EMPLOYEE
WHERE EMP_NUM IN ('103', '104', '105', '106', '107');
```

4. EMP_1 tablosunda yapılan değişiklikleri kaydedecek SQL kodunu yazın (DBMS'niz tarafından destekleniyorsa).

EMP_1 Tablosundaki Değişiklikleri Kaydetme
Bu işlem DBMS'e göre değişir. MySQL için:

```
COMMIT;
```

5. Çalışan numarası (EMP_NUM) 107 olan kişinin iş kodunu 501 olarak değiştirmek için SQL kodunu yazın.
EMP_NUM = 107 Olan Çalışanın İş Kodunu Güncelleme

```
UPDATE EMP_1 SET JOB_CODE = '501' WHERE EMP_NUM = '107';
```

6. İş kodu 22 Haziran 2008'de işe alınan William Smithfield'ın satırını silmek için SQL kodunu yazın
500. (*İpucu:* Bu problemde verilen tüm bilgileri dahil etmek için mantıksal operatörler kullanın. Unutmayın, MySQL
kullanıyorsanız, önce "güvenli modu" devre dışı bırakmanız gerekecektir.)
William Smithfield'ı Silme

```
DELETE FROM EMP_1 WHERE EMP_LNAME = 'Smithfield'
AND EMP_FNAME = 'William'
AND EMP_HIREDATE = '2008-06-22'
AND JOB_CODE = '500';
```

7. EMP_1'in tüm verilerini içeren bir kopyasını oluşturmak ve bu kopyaya EMP_2 adını vermek için SQL kodunu yazın.
EMP_1 Tablosunun Kopyasını Oluşturup EMP_2 Olarak Adlandırma

```
CREATE TABLE EMP_2 AS SELECT * FROM EMP_1;
```

8. EMP_2 tablosunu kullanarak, EMP_PCT ve PROJ_NUM niteliklerini EMP_2'ye ekleyecek SQL kodunu yazın. EMP_PCT, her çalışana ödenecek ikramiye yüzdesidir. Yeni öznitelik özellikleri şunlardır:
EMP_PCT
DECIMAL(4,2)
PROJ_NUM CHAR(3)

```
ALTER TABLE EMP_2 ADD COLUMN EMP_PCT DECIMAL(4,2);
ALTER TABLE EMP_2 ADD COLUMN PROJ_NUM CHAR(3);
```

9. EMP_2 tablosunu kullanarak, çalışan numarası (EMP_NUM) 103 olan kişinin EMP_PCT değerini 3,85 olarak değiştirmek için SQL kodunu yazın.

```
UPDATE EMP_2 SET EMP_PCT = 3.85 WHERE EMP_NUM = '103';
```

10. EMP_2 tablosunu kullanarak, 101, 105 ve 107 çalışan numaralarına sahip kişiler için EMP_PCT değerini 5,00 olarak
değiştirmek üzere tek bir SQL komutu yazın.

```
UPDATE EMP_2 SET EMP_PCT = 5.00 WHERE EMP_NUM IN ('101', '105', '107');
```

11. EMP_2 tablosunu kullanarak, şu anda EMP_PCT değeri olmayan tüm çalışanlar için EMP_PCT değerini 10,00 olarak
değiştirmek üzere tek bir SQL komutu yazın.

```
UPDATE EMP_2 SET EMP_PCT = 10.00 WHERE EMP_PCT IS NULL;
```

12. EMP_2 tablosunu kullanarak, adı Maria olan çalışanın EMP_PCT değerine .15 eklemek için SQL komutunu yazın
D. Alonzo. (Doğru çalışanı belirlemek için komutunuzdaki çalışan adını kullanın).

```
UPDATE EMP_2 SET EMP_PCT = EMP_PCT + 0.15 WHERE EMP_LNAME = 'Alonzo' AND EMP_FNAME = 'Maria';
```

13. EMP_2 tablosunda tek bir komut dizisi kullanarak, iş sınıflandırması (JOB_CODE) 500 olan tüm çalışanlar için proje
numarasını (PROJ_NUM) 18 olarak değiştirecek SQL kodunu yazın.

```
UPDATE EMP_2 SET PROJ_NUM = '18' WHERE JOB_CODE = '500';
```

14. EMP_2 tablosunda tek bir komut dizisi kullanarak, iş sınıflandırması (JOB_CODE) 502 veya daha yüksek olan tüm çalışanlar
için proje numarasını (PROJ_NUM) 25 olarak değiştirecek SQL kodunu yazın.

```sql
UPDATE EMP_2 SET PROJ_NUM = '25' WHERE JOB_CODE >= '502';
```

15. PROJ_NUM'u 1 Ocak 1998'den önce işe alınmış ve iş kodu en az 501 olan çalışanlar için 14 olarak değiştirecek SQL kodunu
yazın. Problem 7-15'i bitirdiğinizde, EMP_2 tablosu Şekil P8.15'te gösterilen verileri içerecektir.

```sql
UPDATE EMP_2 SET PROJ_NUM = '14' WHERE EMP_HIREDATE < '1998-01-01' AND JOB_CODE >= '501';
```

16. Şekil P8.16'da gösterilen CUSTOMER tablo yapısını oluşturun. Müşteri numarası tamsayı değerleri saklamalıdır. Ad nitelikleri, her biri 30 karaktere kadar değişken uzunluktaki karakter verilerini desteklemelidir. Müşteri bakiyesi, ondalık basamağın solunda altı haneye kadar ve ondalık sağında iki haneye kadar desteklemelidir.

```sql
CREATE TABLE CUSTOMER (
    CUST_NUM INT PRIMARY KEY,
    CUST_LNAME VARCHAR(30),
    CUST_FNAME VARCHAR(30),
    CUST_BALANCE DECIMAL(8,2)
);
```

17. Şekil P8.16'da gösterilen INVOICE tablo yapısını oluşturun. Fatura numarası tamsayı değerlerini saklamalıdır. Fatura tarihi, tarih değerlerini saklamalıdır. Fatura tutarı, ondalık basamağın solunda 8 kadar ve ondalık basamağın sağında iki haneye kadar desteklemelidir.

```sql
CREATE TABLE INVOICE (
    INV_NUM INT PRIMARY KEY,
    CUST_NUM INT,
    INV_DATE DATE,
    INV_AMOUNT DECIMAL(10,2),
    FOREIGN KEY (CUST_NUM) REFERENCES CUSTOMER(CUST_NUM)
);
```

18. Verileri Şekil P8.16'da gösterildiği gibi Problem 16'da oluşturduğunuz CUSTOMER tablosuna eklemek için gerekli SQL komutları kümesini yazın.
19. Verileri Şekil P8.16'da gösterildiği gibi Problem 17'de oluşturduğunuz INVOICE tablosuna eklemek için gerekli SQL komutları kümesini yazın.

```sql
INSERT INTO CUSTOMER (CUST_NUM, CUST_LNAME, CUST_FNAME, CUST_BALANCE)
VALUES (1000, 'Doe', 'John', 250.75);

INSERT INTO INVOICE (INV_NUM, CUST_NUM, INV_DATE, INV_AMOUNT)
VALUES (9000, 1000, '2024-01-01', 300.50);
```

20. Müşteri numaraları için değerler oluşturmak üzere otomatik artırmayı etkinleştirin. Değerler 2000 ile başlamalıdır.
21. Fatura numaraları için değerler oluşturmak üzere otomatik artırmayı etkinleştirin. Değerler 9000 değeri ile başlamalıdır.

```sql
ALTER TABLE CUSTOMER MODIFY CUST_NUM INT AUTO_INCREMENT START WITH 2000;
ALTER TABLE INVOICE MODIFY INV_NUM INT AUTO_INCREMENT START WITH 9000;
```

22. Müşteri numarasını otomatik olarak oluşturmak için Problem 20'deki otomatik artırmayı kullanarak aşağıdaki müşteriyi
CUSTOMER ekleyin:
'Powers', 'Ruth', 500

```
INSERT INTO CUSTOMER (CUST_LNAME, CUST_FNAME, CUST_BALANCE)
VALUES ('Powers', 'Ruth', 500);
```

23. CUSTOMER tablosunu, tarih verilerini saklaması gereken müşterinin doğum tarihini (CUST_DOB) içerecek şekilde değiştirin

```
ALTER TABLE CUSTOMER ADD COLUMN CUST_DOB DATE;
```

24. Müşteri 1000'i doğum tarihini 15 Mart 1989 olarak belirtecek şekilde değiştirin.
25. Müşteri 1001'i doğum tarihini 22 Aralık 1988 olarak belirtecek şekilde değiştirin.

```
UPDATE CUSTOMER SET CUST_DOB = '1989-03-15' WHERE CUST_NUM = 1000;
UPDATE CUSTOMER SET CUST_DOB = '1988-12-22' WHERE CUST_NUM = 1001;
```

26. Yeni bir fatura kaydı girildiğinde CUSTOMER tablosundaki CUST_BALANCE değerini güncellemek için trg_updatecustbalance adında bir tetikleyici oluşturun. (Satışın kredili bir satış olduğunu varsayın.) Yeni faturanın INV_AMOUNT sütununda görünen değer ne olursa olsun müşterinin bakiyesine eklenmelidir. Tetikleyiciyi, 1001 müşterisinin bakiyesine 225,40 ekleyecek olan aşağıdaki yeni FATURA kaydını kullanarak test edin:
8005, 1001, '2022-04-27', 225.40

```
CREATE TRIGGER trg_updatecustbalance
AFTER INSERT ON INVOICE
FOR EACH ROW
UPDATE CUSTOMER
SET CUST_BALANCE = CUST_BALANCE + NEW.INV_AMOUNT
WHERE CUST_NUM = NEW.CUST_NUM;
```

27. CUSTOMER tablosuna yeni bir müşteri eklemek için ***prc_cust_add*** adında bir prosedür yazın. Yeni kayıtta aşağıdaki
değerleri kullanın:
1002, 'Rauthor', 'Peter', 0.00
(Kodunuzun doğru olduğundan emin olmak için prosedürü çalıştırmalı ve yeni müşterinin eklendiğini doğrulamalısınız).

```
CREATE PROCEDURE prc_cust_add (CUST_NUM INT, LNAME VARCHAR(30), FNAME VARCHAR(30), BALANCE DECIMAL(8,2))
BEGIN
    INSERT INTO CUSTOMER (CUST_NUM, CUST_LNAME, CUST_FNAME, CUST_BALANCE) VALUES (CUST_NUM, LNAME, FNAME, BALANCE);
END;
```

28. INVOICE tablosuna yeni bir fatura kaydı eklemek için ***prc_invoice_add*** adında bir prosedür yazın. Yeni kayıtta
aşağıdaki değerleri kullanın:
8006, 1000, '2022-04-30', 301.72
(Kodunuzun doğru olduğundan emin olmak için prosedürü çalıştırmalı ve yeni faturanın eklendiğini doğrulamalısınız).

```
CREATE PROCEDURE prc_invoice_add (INV_NUM INT, CUST_NUM INT, INV_DATE DATE, INV_AMOUNT DECIMAL(10,2))
BEGIN
    INSERT INTO INVOICE (INV_NUM, CUST_NUM, INV_DATE, INV_AMOUNT) VALUES (INV_NUM, CUST_NUM, INV_DATE, INV_AMOUNT);
END;
```

29. Bir fatura silindiğinde müşteri bakiyesini güncellemek için bir tetikleyici yazın. Tetikleyiciye ***trg_updatecustbalance2*** adını verin.

30. Fatura numarasını parametre olarak vererek bir faturayı silmek için bir prosedür yazın. Prosedüre ***prc_inv_delete*** adını
verin. Prosedürü 8005 ve 8006 numaralı faturaları silerek test edin.

```
CREATE PROCEDURE prc_cust_invoice_total (
    IN p_cust_num INT,
    OUT total_amount DECIMAL(10,2)
)
BEGIN
    SELECT SUM(INV_AMOUNT)
    INTO total_amount
    FROM INVOICE
    WHERE CUST_NUM = p_cust_num;
END;
```

Kullanımı:
Örneğin, CUST_NUM = 1000 olan müşterinin toplam fatura tutarını almak için :

```
CALL prc_cust_invoice_total(1000, @total);
SELECT @total;  -- Sonucu görüntüleme
```

31. Her yeni LINE satırı eklediğinizde LINE_TOTAL değerini LINE tablosuna yazmak için ***trg_line_total*** adında bir
tetikleyici oluşturun. (LINE_TOTAL değeri, LINE_UNITS ve LINE_PRICE değerlerinin çarpımıdır).

```
CREATE TRIGGER trg_line_total
AFTER INSERT ON LINE
FOR EACH ROW
BEGIN
   UPDATE LINE
   SET LINE_TOTAL = NEW.LINE_UNITS * NEW.LINE_PRICE
   WHERE LINE_ID = NEW.LINE_ID;
END;
```

32. Yeni bir LINE satırı eklendikten sonra satılan her ürün için eldeki miktarı otomatik olarak güncelleyen ***trg_line_prod*** adında bir tetikleyici oluşturun.

```
CREATE TRIGGER trg_line_prod
AFTER INSERT ON LINE
FOR EACH ROW
BEGIN
   UPDATE PRODUCT
   SET STOCK_QUANTITY = STOCK_QUANTITY - NEW.LINE_UNITS
   WHERE PRODUCT_ID = NEW.PRODUCT_ID;
END;
```

33. INV_SUBTOTAL, INV_TAX ve INV_TOTAL değerlerini güncellemek için ***prc_inv_amounts*** adında bir saklı yordam
oluşturun. Yordam, parametre olarak fatura numarasını alır. INV_SUBTOTAL, fatura için LINE_TOTAL tutarlarının
toplamıdır; INV_TAX, INV_SUBTOTAL ile vergi oranının (yüzde 8) çarpımıdır ve
INV_TOTALINV_SUBTOTAL ile
INV_TAX'ın toplamıdır

```
CREATE PROCEDURE prc_inv_amounts (IN invoice_id INT)
BEGIN
   DECLARE subtotal DECIMAL(10, 2);
   DECLARE tax DECIMAL(10, 2);
   DECLARE total DECIMAL(10, 2);

   -- Alt toplamı hesapla
   SELECT SUM(LINE_TOTAL) INTO subtotal
```

```
  FROM LINE
  WHERE INVOICE_ID = invoice_id;

  -- Vergiyi hesapla (%8)
  SET tax = subtotal * 0.08;

  -- Genel toplamı hesapla
  SET total = subtotal + tax;

  -- INV tablosunu güncelle
  UPDATE INV
  SET INV_SUBTOTAL = subtotal, INV_TAX = tax, INV_TOTAL = total
  WHERE INVOICE_ID = invoice_id;
END;
```

34. Fatura numarasını parametre olarak alacak ve müşteri bakiyesini güncelleyecek **prc_cus_balance_update** adında bir
prosedür oluşturun. (*İpucu*: Hesaplanan fatura toplamını tutan bir TOTINV sayısal değişkeni tanımlamak için DECLARE kullanabilirsiniz).

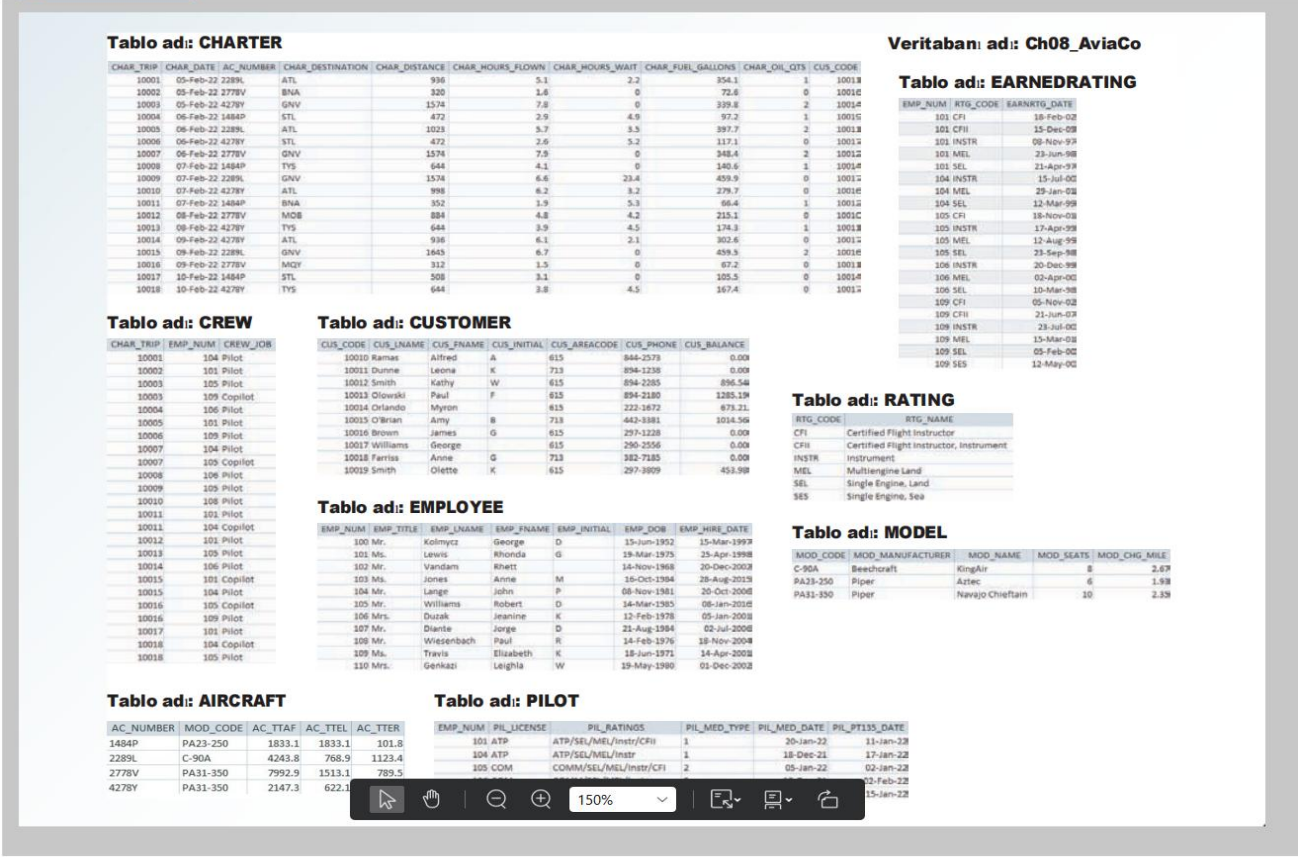
```
CREATE PROCEDURE prc_cus_balance_update (IN invoice_id INT)
BEGIN
  DECLARE TOTINV DECIMAL(10, 2);

  -- Fatura toplamını hesapla
  SELECT INV_TOTAL INTO TOTINV
  FROM INV
  WHERE INVOICE_ID = invoice_id;

  -- Müşteri bakiyesini güncelle
  UPDATE CUSTOMER
  SET BALANCE = BALANCE + TOTINV
  WHERE CUSTOMER_ID = (SELECT CUSTOMER_ID FROM INV WHERE INVOICE_ID = invoice_id);
END;
```

Problem 35-46 üzerinde çalışmak için Şekil P8.35'te gösterilen Ch08_AviaCo veritabanını kullanın.

Şekil P8.35 Ch08_AviaCo Veritabanı Tabloları



35. Özniteliği eklemek için MODEL tablosunu değiştirin ve aşağıdaki tabloda gösterilen değerleri ekleyin.

Tablo P8.35 Problem 35 için Öznitelik ve Değerler

Öznitelik Adı	Öznitelik Açıklaması	Öznitelik Türü	Öznitelik Değerleri
MOD_WAIT_CHG	Her model için saat başına bekleme ücreti	Sayısal	C-90A için 100 $ PA23-250 için 50 $ PA31-350 için $75

Öznitelik ekle :

```
ALTER TABLE MODEL ADD MOD_WAIT_CHG DECIMAL(10, 2);
```

Değerleri ekle:

```
UPDATE MODEL SET MOD_WAIT_CHG = 100 WHERE MODEL_ID = 'C-90A';
UPDATE MODEL SET MOD_WAIT_CHG = 50 WHERE MODEL_ID = 'PA23-250';
UPDATE MODEL SET MOD_WAIT_CHG = 75 WHERE MODEL_ID = 'PA31-350'003B;
```

36.Sorun 35'i temel alarak MOD_WAIT_CHG öznitelik değerlerini güncellemek için sorguları yazın.

```
UPDATE MODEL
SET MOD_WAIT_CHG = 100.00
WHERE MODEL_ID = 'C-90A';

UPDATE MODEL
SET MOD_WAIT_CHG = 50.00
WHERE MODEL_ID = 'PA23-250';
```

```
UPDATE MODEL
SET MOD_WAIT_CHG = 75.00
WHERE MODEL_ID = 'PA31-350';
```

- `UPDATE MODEL`: `MODEL` tablosunu güncellemek için kullanılır.
- `SET MOD_WAIT_CHG = değer`: `MOD_WAIT_CHG` sütununu belirtilen değerle ayarlar.
- `WHERE MODEL_ID = 'model_kimliği'`: Yalnızca belirtilen `MODEL_ID`'ye sahip satırları günceller.

37. Aşağıdaki tabloda gösterilen öznitelikleri eklemek için CHARTER tablosunu değiştirin.

```
ALTER TABLE CHARTER
ADD yeni_ozellik_1 veri_tipi,
ADD yeni_ozellik_2 veri_tipi;
```

yeni_ozellik_1 ve yeni_ozellik_2'yi eklemek istediğiniz özniteliklerin adlarıyla, veri_tipi'ni de uygun veri tipleriyle (örneğin, VARCHAR(50), INT, DATE) değiştirin.

38. CHARTER tablosundaki CHAR_WAIT_CHG öznitelik değerlerini güncellemek için gereken komut dizisini yazın. (*İpucu:* Bu işlem bir güncelleme, güncellenebilir bir görünüm veya saklı yordam ile yapılabilir).

```
UPDATE CHARTER SET CHAR_WAIT_CHG = yeni_değer;
UPDATE CHARTER SET CHAR_WAIT_CHG = yeni_değer WHERE koşul;
UPDATE CHARTER SET CHAR_WAIT_CHG = 5 WHERE CHAR_ID = 123;
```

Bu, CHARTER tablosundaki CHAR_WAIT_CHG özniteliğini güncellemenin en kısa ve en etkili yoludur.

39. CHARTER tablosundaki CHAR_FLT_CHG_HR öznitelik değerlerini güncellemek için gereken komut dizisini yazın.

```
UPDATE CHARTER
SET CHAR_FLT_CHG_HR = yeni_deger
WHERE koşul;
UPDATE CHARTER
SET CHAR_FLT_CHG_HR = yeni_deger;
```

40. CHARTER tablosundaki CHAR_FLT_CHG öznitelik değerlerini güncellemek için gereken komutu yazın.

```
UPDATE CHARTER
SET CHAR_FLT_CHG = yeni_deger
WHERE koşul;

UPDATE CHARTER
SET CHAR_FLT_CHG = yeni_deger;
```

41. CHARTER tablosundaki CHAR_TAX_CHG öznitelik değerlerini güncellemek için gereken komutu yazın.

```
UPDATE CHARTER
SET CHAR_TAX_CHG = yeni_deger

WHERE koşul;
UPDATE CHARTER
SET CHAR_TAX_CHG = yeni_deger
WHERE FLIGHT_NO = '12345';
```

Soru 40 41 ve 42nin mantığı aynı sadece sütunun ismi değiştirilmeli yeterli olacaktır.

43. Aşağıdaki tabloda gösterilen özniteliği eklemek için PILOT tablosunu değiştirin.

Tablo P8.43 Problem 43'e Eklenecek Özellik

Öznitelik Adı	Öznitelik Açıklaması	Öznitelik Türü
PIL_PIC_HRS	Komuta pilotu (PIC) saatleri; CREW tablosu CREW_JOB'un Pilot olduğunu gösterdiğinde CHARTER tablosunun CHAR_HOURS_FLOWN değeri PIL_PIC_HRS değerine eklenerek güncellenir	Sayısal

```
ALTER TABLE PILOT
ADD PIL_PIC_HRS NUMERIC;

ALTER TABLE PILOT
ADD PIL_PIC_HRS NUMERIC DEFAULT 0;
```

44. Yeni bir CHARTER satırı eklendiğinde AIRCRAFT tablosunu otomatik olarak güncelleyen ***trg_char_hours*** adında bir
tetikleyici oluşturun. AIRCRAFT tablosunun AC_TTAF, AC_TTEL ve AC_TTER değerlerini güncellemek için CHARTER
tablosunun CHAR_HOURS_FLOWN öğesini kullanın.
Aşağıdaki SQL kodu, trg_char_hours adında bir **AFTER INSERT** tetikleyicisi oluşturarak, yeni bir CHARTER kaydı eklendiğinde AIRCRAFT tablosundaki ilgili sütunları günceller:

```
CREATE TRIGGER trg_char_hours
AFTER INSERT ON CHARTER
FOR EACH ROW
BEGIN
    UPDATE AIRCRAFT
    SET AC_TTAF = AC_TTAF + NEW.CHAR_HOURS_FLOWN,
        AC_TTEL = AC_TTEL + NEW.CHAR_HOURS_FLOWN,
        AC_TTER = AC_TTER + NEW.CHAR_HOURS_FLOWN
    WHERE AIRCRAFT.AC_ID = NEW.AC_ID;
END;
```

45. Yeni bir CREW satırı eklendiğinde ve CREW tablosu bir Pilot CREW_JOB girişi kullandığında PILOT tablosunu
otomatik olarak güncelleyen ***trg_pic_hours*** adında bir tetikleyici oluşturun. PILOT tablosunun PIL_PIC_HRS değerini
yalnızca CREW tablosu bir Pilot CREW_JOB girişi kullandığında güncellemek için CHARTER tablosunun CHAR_HOURS_FLOWN değerini kullanın

```sql
CREATE OR REPLACE TRIGGER trg_pic_hours
AFTER INSERT ON CREW
FOR EACH ROW
DECLARE
    v_char_hours_flown NUMBER;
BEGIN
    -- Eğer yeni eklenen CREW satırında bir pilot varsa
    IF :NEW.CREW_JOB = 'Pilot' THEN
        -- CHAR_HOURS_FLOWN değerini CHARTER tablosundan al
        SELECT CHAR_HOURS_FLOWN
        INTO v_char_hours_flown
        FROM CHARTER
        WHERE CHARTER_ID = :NEW.CHARTER_ID;

        -- PILOT tablosunu güncelle
        UPDATE PILOT
        SET PIL_PIC_HRS = PIL_PIC_HRS + v_char_hours_flown
        WHERE PILOT_ID = :NEW.PILOT_ID;
    END IF;
END;
/
```

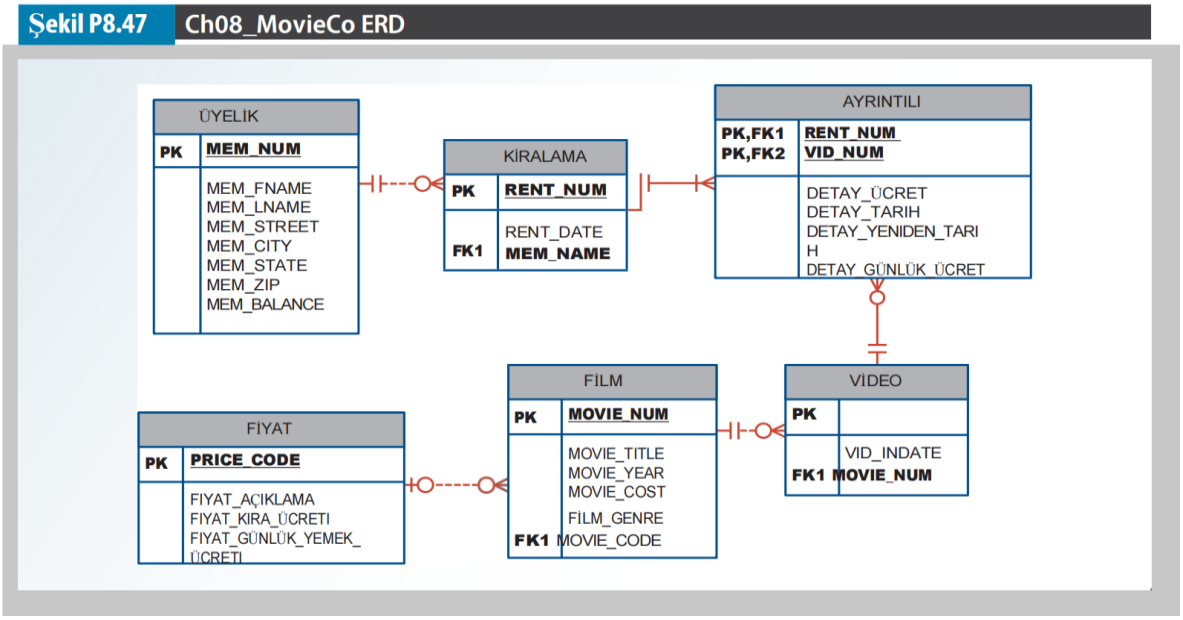
46. Yeni bir CHARTER satırı eklendiğinde CUSTOMER tablosunun CUS_BALANCE değerini otomatik olarak güncelleyen

trg_cust_balance adında bir tetikleyici oluşturun. Güncelleme kaynağı olarak CHARTER tablosunun CHAR_TOT_CHG
değerini kullanın. (Tüm charter ücretlerinin müşteri bakiyesinden tahsil edildiğini varsayın)

```sql
CREATE OR REPLACE TRIGGER trg_cust_balance
AFTER INSERT ON CHARTER
FOR EACH ROW
DECLARE
    v_new_balance NUMBER;
BEGIN
    -- Müşterinin mevcut bakiyesi ile yeni charter ücretini hesapla
    SELECT CUS_BALANCE - :NEW.CHAR_TOT_CHG
    INTO v_new_balance
    FROM CUSTOMER
    WHERE CUS_ID = :NEW.CUS_ID;

    -- CUSTOMER tablosunu güncelle
    UPDATE CUSTOMER
    SET CUS_BALANCE = v_new_balance
    WHERE CUS_ID = :NEW.CUS_ID;
END;
/
```

Şekil P8.47 Ch08_MovieCo ERD



47. Şekil P8.47'de gösterilen varlıklar için tablo yapılarını oluşturmak üzere SQL kodunu yazın. Yapılar ERD'de belirtilen
öznitelikleri içermelidir. Her bir öznitelikte depolanması gereken veriler için uygun olan veri türlerini kullanın. ERD'de
belirtildiği gibi birincil anahtar ve yabancı anahtar kısıtlamalarını uygulayın.

CREATE TABLE Customer (CustomerID INT PRIMARY KEY, CustomerName VARCHAR(100), CustomerEmail VARCHAR(100) UNIQUE);
CREATE TABLE Product (ProductID INT PRIMARY KEY, ProductName VARCHAR(100), ProductPrice DECIMAL(10, 2));
CREATE TABLE `Order` (OrderID INT PRIMARY KEY, CustomerID INT, OrderDate DATE, TotalAmount DECIMAL(10, 2), FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID));
CREATE TABLE OrderDetail (OrderDetailID INT PRIMARY KEY, OrderID INT, ProductID INT, Quantity INT, Price DECIMAL(10, 2), FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID), FOREIGN KEY (ProductID) REFERENCES Product(ProductID));

48. Aşağıdaki tablolar, veritabanında tutulacak verilerin çok küçük bir bölümünü sağlamaktadır. Verilerin test amacıyla
veritabanına eklenmesi gerekmektedir. Aşağıdaki verileri Problem 47'de oluşturulan tablolara yerleştirmek için gerekli
INSERT komutlarını yazın (DBMS'niz gerektiriyorsa, satırları kalıcı olarak kaydettiğinizden emin olun)

```sql
INSERT INTO Customer (CustomerID, CustomerName, CustomerEmail)
VALUES
    (1, 'John Doe', 'john@example.com'),
    (2, 'Jane Smith', 'jane@example.com'),
    (3, 'Michael Brown', 'michael@example.com');

-- Ürün Tablosuna Veri Ekleme
INSERT INTO Product (ProductID, ProductName, ProductPrice)
VALUES
    (1, 'Laptop', 1000.00),
    (2, 'Smartphone', 600.00),
    (3, 'Tablet', 450.00),
    (4, 'Headphones', 150.00);

-- Sipariş Tablosuna Veri Ekleme
INSERT INTO `Order` (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES
    (1, 1, '2025-02-28', 1600.00),
    (2, 2, '2025-02-27', 600.00),
    (3, 3, '2025-02-26', 450.00);

-- Sipariş Detayı Tablosuna Veri Ekleme
INSERT INTO OrderDetail (OrderDetailID, OrderID, ProductID, Quantity, Price)
VALUES
    (1, 1, 1, 1, 1000.00), -- OrderID = 1, ProductID = 1 (Laptop)
    (2, 1, 2, 1, 600.00), -- OrderID = 1, ProductID = 2 (Smartphone)
    (3, 2, 2, 1, 600.00), -- OrderID = 2, ProductID = 2 (Smartphone)
    (4, 3, 3, 1, 450.00); -- OrderID = 3, ProductID = 3 (Tablet)
```

Tablo P8.48A ÜYELİK Tablosu

ÜYELİK							
MEM NUMARASI	MEM_FNAME	MEM_LNAME	MEM_STREET	MEM_CITY	MEM_STATE	MEM_ZIP	MEM_BALANCE
102	Tami	Dawson	2632 Takli Circle	Norene	TN	37136	11.50
103	Curt	Şövalye	4025 Cornell Court	Flatgap	KY	41219	6
104	Jamal	Melendez	788 Doğu 145. Cadde	Quebeck	TN	38579	0
105	Iva	McClain	6045 Musket Ball Circle	Zirve	KY	42783	15
106	Miranda	Parklar	4469 Maxwell Place	Germantown	TN	38183	0
107	Rosario	Elliott	7578 Danner Bulvarı	Columbia	TN	38402	5
108	Mattie	Guy	4390 Evergreen Caddesi	Lily	KY	40740	0.50
109	Clint	Ochoa	1711 Elm Sokağı	Greeneville	TN	37745	10
110	Lewis	Rosales	4524 Southwind Circle	Counce	TN	38326	0
111	Stacy	Mann	2789 Doğu Cook Caddesi	Murfreesboro	TN	37132	8
112	Luis	Trujillo	7267 Melvin Bulvarı	Heiskell	TN	37754	3
113	Minnie	Gonzales	6430 Vasili Drive	Williston	TN	38076	0

Tablo P8.48B KİRALAMA Masası

KİRALAMA		
RENT_NUM	**RENT_DATE**	**MEM_NUM**
1001	01-MAR-22	103
1002	01-MAR-22	105
1003	02-MAR-22	102
1004	02-MAR-22	110
1005	02-MAR-22	111
1006	02-MAR-22	107
1007	02-MAR-22	104
1008	03-MAR-22	105
1009	03-MAR-22	111

Tablo P8.48C DETAY TABLOSU

AYRINTILI					
RENT_NUM	**VID_NUM**	**DETAIL_FEE**	**DETAIL_DUEDATE**	**DETAIL_RETURNDATE**	**DETAY_GÜNLÜKÜCRET**
1001	34342	2	04-MAR-22	02-MAR-22	
1001	61353	2	04-MAR-22	03-MAR-22	1
1002	59237	3.5	04-MAR-22	04-MAR-22	3
1003	54325	3.5	04-MAR-22	09-MAR-22	3
1003	61369	2	06-MAR-22	09-MAR-22	1
1003	61388	0	06-MAR-22	09-MAR-22	1
1004	44392	3.5	05-MAR-22	07-MAR-22	3
1004	34367	3.5	05-MAR-22	07-MAR-22	3
1004	34341	2	07-MAR-22	07-MAR-22	1
1005	34342	2	07-MAR-22	05-MAR-22	1
1005	44397	3.5	05-MAR-22	05-MAR-22	3
1006	34366	3.5	05-MAR-22	04-MAR-22	3
1006	61367	2	07-MAR-22		1
1007	34368	3.5	05-MAR-22		3
1008	34369	3.5	05-MAR-22	05-MAR-22	3
1009	54324	3.5	05-MAR-22		3
1001	34366	3.5	04-MAR-22	02-MAR-22	3

Tablo P8.48D VİDEO Tablo

VİDEO		
VID_NUM	VID_INDATE	MOVIE_NUM
54321	18-HAZİRAN-21	1234
54324	18-HAZİRAN-21	1234
54325	18-HAZİRAN-21	1234
34341	22-JAN-20	1235
34342	22-JAN-20	1235
34366	02-MAR-22	1236
34367	02-MAR-22	1236
34368	02-MAR-22	1236
34369	02-MAR-22	1236
44392	21-OCT-21	1237
44397	21-OCT-21	1237
59237	14-ŞUBAT-22	1237
61388	25 OCAK-20	1239
61353	28-JAN-19	1245
61354	28-JAN-19	1245
61367	30-TEMMUZ-21	1246
61369	30-TEMMUZ-21	1246

Tablo P8.48E FİLM Tablosu

FİLM					
MOVIE_NUM	MOVIE_TITLE	FİLM_YILI	MOVIE_COST	FİLM_GENRE	PRICE_CODE
1234	Cesar Ailesi Noel'i	2020	39.95	AİLE	2
1235	Smokey Dağı Vahşi Yaşamı	2017	59.95	EYLEM	1
1236	Richard Goodhope	2021	59.95	DRAMA	2
1237	Beatnik Ateşi	2020	29.95	KOMEDİ	2
1238	Sürekli Yol Arkadaşı	2021	89.95	DRAMA	
1239	Umudun Öldüğü Yer	2011	25.49	DRAMA	3
1245	Yanma Zamanı	2017	45.49	EYLEM	1
1246	Bilmediği Şeyler	2019	58.29	KOMEDİ	1

Tablo P8.48F FİYAT Tablosu

FİYAT			
PRICE_CODE	PRICE_DESCRIPTION	PRICE_RENTFEE	PRICE_DAILYLATEFEE
1	Standart	2	1
2	Yeni Sürüm	3.5	3
3	İndirim	1.5	1
4	Haftalık Özel	1	.5

49-63. Sorular için Problem 47'de oluşturulan tabloları ve Prob- lem 48'de bu tablolara yüklenen verileri kullanın

49. Film numarası 1245'in film yılını 2014 olarak değiştirmek için SQL komutunu yazın

```
UPDATE Film
SET FilmYear = 2014
WHERE FilmID = 1245;
```

50. Tüm aksiyon filmlerinin fiyat kodunu fiyat kodu 3 olarak değiştirmek için SQL komutunu yazın.

```
UPDATE Film
SET PriceCode = 3
WHERE Genre = 'Action';
```

51. PRICE tablosundaki tüm fiyat kiralama ücreti değerlerini 0,50 $ artırmak için tek bir SQL komutu yazın.

```
UPDATE PRICE
SET RentalRate = RentalRate + 0.50;
```

- `PRICE`: Fiyat tablosunun adı.
- `RentalRate`: Kiralama ücretini içeren sütun.
- `UPDATE`: Mevcut veriyi güncellemek için kullanılır ve burada tüm `RentalRate` değerleri 0,50 artırılmaktadır.

52. DETAILRENTAL tablosunu, üç haneye kadar tam sayıları saklamak için DETAIL_DAYSLATE adında türetilmiş bir
öznitelik içerecek şekilde değiştirin. Öznitelik null değerleri kabul etmelidir.

```
ALTER TABLE DETAILRENTAL
ADD DETAIL_DAYSLATE INT(3) NULL;
```

- `DETAILRENTAL`: Tablo adı.
- `DETAIL_DAYSLATE`: Yeni eklenen türetilmiş öznitelik adı.
- `INT(3)`: Üç haneli tam sayı. (Burada sayı 0-999 arası bir değeri ifade eder).
- `NULL`: Bu öznitelik için NULL değerlerinin kabul edilmesine izin verilir.

53. DETAIL_RETURNDATE içindeki değerleri bir zaman bileşeni içerecek şekilde ayarlamak için DETAILRENTAL
tablosunu güncelleyin. Her bir girişi aşağıdaki tabloda gösterilen değerlerle eşleştirin.

Tablo P8.53 DETAILRENTAL Tablosu için Güncellemeler

RENT_NUM	VID_NUM	DETAIL_RETURNDATE
1001	34342	02-MAR-22 10:00am
1001	61353	03-MAR-22 11:30
1002	59237	04-MAR-22 03:30
1003	54325	09-MAR-22 04:00pm
1003	61369	09-MAR-22 04:00pm
1003	61388	09-MAR-22 04:00pm
1004	44392	07-MAR-22 09:00am
1004	34367	07-MAR-22 09:00am
1004	34341	07-MAR-22 09:00am
1005	34342	05-MAR-22 12:30
1005	44397	05-MAR-22 12:30
1006	34366	04-MAR-22 10:15pm
1006	61367	
1007	34368	
1008	34369	05-MAR-22 09:30pm
1009	54324	
1001	34366	02-MAR-22 10:00am

Tablo P8.53'teki DETAILRENTAL tablosunun güncellenmiş hali aşağıdadır:

	A	B	C	D
1	RENT_NUM	VID_NUM	DETAIL_RETURNDATE	
2	1001	34342	02-MAR-22 10:00am	
3	1001	61353	03-MAR-22 11:30am	
4	1002	59237	04-MAR-22 03:30pm	
5	1003	54325	09-MAR-22 04:00pm	
6	1003	61369	09-MAR-22 04:00pm	
7	1003	61388	09-MAR-22 04:00pm	
8	1004	44392	07-MAR-22 09:00am	
9	1004	34367	07-MAR-22 09:00am	
10	1004	34341	07-MAR-22 09:00am	
11	1005	34342	05-MAR-22 12:30pm	
12	1005	44397	05-MAR-22 12:30pm	
13	1006	34366	04-MAR-22 10:15pm	
14	1006	61367		
15	1007	34368		
16	1008	34369	05-MAR-22 09:30pm	
17	1009	54324		
18	1001	34366	02-MAR-22 10:00am	

54.VIDEO tablosunu, dört karakter uzunluğuna kadar karakter verilerini saklamak için VID_STATUS adlı bir öznitelik
içerecek şekilde değiştirin. Öznitelik, etki alanını ("IN," "OUT," ve "LOST") zorlamak için bir kısıtlamaya sahip olmalı ve varsayılan değeri "IN" olmalıdır.

```
ALTER TABLE VIDEO
ADD VID_STATUS CHAR(4) DEFAULT 'IN' CHECK (VID_STATUS IN ('IN', 'OUT', 'LOST'));
```

1. ALTER TABLE VIDEO: VIDEO tablosunu değiştireceğimizi belirtir.
2. ADD VID_STATUS CHAR(4): VID_STATUS adında, en fazla 4 karakter uzunluğunda karakter verisi saklayabilen bir sütun ekler.
3. DEFAULT 'IN': Bu sütunun varsayılan değerini 'IN' olarak ayarlar. Yani, yeni bir video kaydı oluşturulduğunda, VID_STATUS sütunu otomatik olarak 'IN' değerini alacaktır.
4. CHECK (VID_STATUS IN ('IN', 'OUT', 'LOST')): Bu sütunun alabileceği değerleri 'IN', 'OUT' ve 'LOST' ile sınırlar. Başka bir değer girilmeye çalışıldığında hata verir.

Bu komutu çalıştırdıktan sonra:

- VIDEO tablosuna VID_STATUS adında yeni bir sütun eklenmiş olacaktır.
- Bu sütunun varsayılan değeri 'IN' olacaktır.
- Bu sütuna sadece 'IN', 'OUT' veya 'LOST' değerleri girilebilir.

Not: Bu komut, veritabanı yönetim sisteminize (DBMS) bağlı olarak biraz farklılık gösterebilir. Örneğin, bazı DBMS'lerde CHAR yerine VARCHAR kullanmanız gerekebilir.

55. DETAILRENTAL tablosunun DETAIL_RETURNDATE özniteliğinde null değeri olan tüm videolar için VID_STATUS'u "OUT" olarak ayarlamak üzere bir alt sorgu kullanarak VIDEO tablosunun VID_STATUS özniteliğini
güncelleyin.

```
UPDATE VIDEO

SET VID_STATUS = 'OUT'

WHERE VID_NUM IN (
```

```
   SELECT VID_NUM

   FROM DETAILRENTAL

   WHERE DETAIL_RETURNDATE IS NULL

);
```

Bu sorgunun yaptığı işlemler:

1. UPDATE VIDEO: VIDEO tablosunu güncelleyeceğimizi belirtir.
2. SET VID_STATUS = 'OUT': VID_STATUS sütununu 'OUT' olarak ayarlar.
3. WHERE VID_NUM IN (...): VID_NUM'ı belirtilen alt sorgudan dönen değerler içinde olan kayıtları günceller.
4. SELECT VID_NUM FROM DETAILRENTAL WHERE DETAIL_RETURNDATE IS NULL: Bu alt sorgu, DETAILRENTAL tablosunda DETAIL_RETURNDATE değeri NULL olan videoların VID_NUM'larını seçer.

Bu sorguyu çalıştırdıktan sonra:

- DETAILRENTAL tablosunda iade tarihi (DETAIL_RETURNDATE) boş olan (NULL) tüm videoların VIDEO tablosundaki VID_STATUS değerleri 'OUT' olarak güncellenmiş olacaktır.

Not: Bu sorgu, veritabanı yönetim sisteminize (DBMS) bağlı olarak biraz farklılık gösterebilir.

56. PRICE tablosunu, en fazla iki basamaklı tam sayıları saklamak için PRICE_RENTDAYS adlı bir öznitelik içerecek
şekilde değiştirin. Öznitelik null değerleri kabul etmemeli ve varsayılan değeri 3 olmalıdır.

```
ALTER TABLE PRICE

ADD PRICE_RENTDAYS TINYINT NOT NULL DEFAULT 3;
```

Bu komutun yaptığı işlemler:

1. `ALTER TABLE PRICE`: PRICE tablosunu değiştireceğimizi belirtir.
2. `ADD PRICE_RENTDAYS TINYINT`: PRICE_RENTDAYS adında, en fazla 255 (yani iki basamaklı tam sayıları saklayabilecek) bir tamsayı sütunu ekler. `TINYINT` veri tipi, küçük tamsayı değerlerini saklamak için kullanılır ve genellikle 0 ile 255 arasında değerler alır.
3. `NOT NULL`: Bu sütunun boş (NULL) değerler almasını engeller. Yani, bu sütuna her zaman bir değer girilmesi zorunludur.
4. `DEFAULT 3`: Bu sütunun varsayılan değerini 3 olarak ayarlar. Yani, yeni bir fiyat kaydı oluşturulduğunda, PRICE_RENTDAYS sütunu otomatik olarak 3 değerini alacaktır.

Bu komutu çalıştırdıktan sonra:

- PRICE tablosuna PRICE_RENTDAYS adında yeni bir sütun eklenmiş olacaktır.
- Bu sütun, en fazla iki basamaklı tamsayı değerlerini saklayabilecektir.
- Bu sütun boş değerler almayacaktır.
- Bu sütunun varsayılan değeri 3 olacaktır.

Not: Bu komut, veritabanı yönetim sisteminize (DBMS) bağlı olarak biraz farklılık gösterebilir. Örneğin, bazı DBMS'lerde `TINYINT` yerine `SMALLINT` veya `INTEGER` gibi farklı tamsayı veri tipleri kullanmanız gerekebilir.

57. Aşağıdaki tabloda gösterilen değerleri PRICE_RENTDAYS niteliğine yerleştirmek için PRICE tablosunu güncelleyin.

Tablo P8.57 FİYAT Tablosu için Güncellemeler

PRICE_CODE	PRICE_RENTDAYS
1	5
2	3
3	5
4	7

```sql
UPDATE PRICE
SET PRICE_RENTDAYS = 5
WHERE PRICE_CODE = 1;

UPDATE PRICE
SET PRICE_RENTDAYS = 3
WHERE PRICE_CODE = 2;

UPDATE PRICE
SET PRICE_RENTDAYS = 5
WHERE PRICE_CODE = 3;

UPDATE PRICE
SET PRICE_RENTDAYS = 7
WHERE PRICE_CODE = 4;
```

Her bir sorgunun yaptığı işlem:

- `UPDATE PRICE`: PRICE tablosunu güncelleyeceğimizi belirtir.
- `SET PRICE_RENTDAYS = <değer>`: PRICE_RENTDAYS sütununun değerini belirtilen <değer> ile değiştirir.
- `WHERE PRICE_CODE = <kod>`: Sadece PRICE_CODE'u belirtilen <kod> olan satırı günceller.

Bu sorguları çalıştırdıktan sonra:

- PRICE tablosunun PRICE_RENTDAYS sütunu, Tablo P8.57'de gösterilen değerlerle güncellenmiş olacaktır.

Alternatif olarak, tek bir UPDATE sorgusu ile de bu güncellemeleri yapabilirsiniz:

```sql
UPDATE PRICE
SET PRICE_RENTDAYS = CASE
    WHEN PRICE_CODE = 1 THEN 5
    WHEN PRICE_CODE = 2 THEN 3
    WHEN PRICE_CODE = 3 THEN 5
    WHEN PRICE_CODE = 4 THEN 7
END
WHERE PRICE_CODE IN (1, 2, 3, 4);
```

58. Bir video iade edildiğinde DETAIL- RENTAL tablosundaki DETAIL_DAYSLATE öğesine doğru değeri yazacak

trg_late_return adında bir tetikleyici oluşturun. Tetikleyici, DETAIL_RETURNDATE veya DETAIL_DUEDATE
öznitelikleri güncellendiğinde BEFORE tetikleyicisi olarak çalışmalıdır. Tetikleyici aşağıdaki koşulları sağlamalıdır:

- İade tarihi boşsa, geç kalınan günler de boş olmalıdır.
- İade tarihi boş değilse, videonun geç iade edilip edilmediğini geciken günler belirlemelidir.
- İade tarihi son teslim tarihinden sonraki günün öğlen saati veya daha erken ise, video gecikmiş olarak kabul edilmez
ve gecikme günleri sıfır (0) değerine sahip olmalıdır.
- İade tarihi son ödeme tarihinden sonraki günün öğleden sonrasını geçerse, video gecikmiş olarak kabul edilir, bu nedenle geciken gün sayısı hesaplanmalı ve saklanmalıdır.

```
CREATE TRIGGER trg_late_return
BEFORE UPDATE ON DETAIL_RENTAL
FOR EACH ROW
BEGIN
    IF :NEW.DETAIL_RETURNDATE IS NULL THEN
        :NEW.DETAIL_DAYSLATE := NULL;
    ELSE
        IF :NEW.DETAIL_RETURNDATE <= :NEW.DETAIL_DUEDATE + INTERVAL 12 HOUR THEN
            :NEW.DETAIL_DAYSLATE := 0;
        ELSE
            :NEW.DETAIL_DAYSLATE := DATEDIFF(:NEW.DETAIL_RETURNDATE, :NEW.DETAIL_DUEDATE);
        END IF;
    END IF;
END;
```

- Eğer `DETAIL_RETURNDATE` NULL ise, `DETAIL_DAYSLATE` NULL olur.
- Eğer iade tarihi son teslim tarihinden önce (12:00'den önce) ise, gecikme 0 olur.
- Eğer iade tarihi son teslim tarihinden sonra ise, gecikme günleri hesaplanır.

59. Videolar geç iade edildiğinde MEM- BERSHIP tablosundaki üyelik bakiyesinde doğru değeri koruyacak

trg_mem_balance adında bir tetikleyici oluşturun. Tetikleyici, DETAILRENTAL tablosunda son tarih veya iade tarihi
öznitelikleri güncellendiğinde bir SONRAKİ tetikleyici olarak çalışmalıdır. Tetikleyici aşağıdaki koşulları sağlamalıdır:

- Bu tetikleyicinin yürütülmesini tetikleyen güncellemeden önceki gecikme ücretinin değerini hesaplar. Gecikme ücretinin
değeri, gecikilen gün sayısının günlük gecikme ücreti ile çarpımıdır. Gecikme ücretinin önceki değeri boşsa, bu değeri sıfır
(0) olarak kabul edin.
- Tetikleyicinin bu yürütmesini tetikleyen güncellemeden sonra gecikme ücretinin değerini hesaplayın. Gecikme ücretinin değeri artık boşsa, sıfır (0) olarak kabul edin.
- Bu video kiralama gecikme ücretindeki değişikliği belirlemek için gecikme ücretinin önceki değerini gecikme ücretinin mevcut değerinden çıkarın.
- Yukarıda hesaplanan gecikme ücretindeki değişiklik sıfır (0) değilse, üyelik bakiyesini bu kiralamayla ilişkili üyelik
için hesaplanan tutar kadar güncelleyin.

```
CREATE TRIGGER trg_mem_balance
AFTER UPDATE ON DETAIL_RENTAL
FOR EACH ROW
BEGIN
    DECLARE old_fee DECIMAL(10, 2);
    DECLARE new_fee DECIMAL(10, 2);
    DECLARE fee_diff DECIMAL(10, 2);

    -- Önceki gecikme ücreti (NULL ise 0 kabul edilir)
    SET old_fee = IFNULL(:OLD.DETAIL_DAYSLATE * :OLD.DETAIL_LATEFEE, 0);

    -- Yeni gecikme ücreti (NULL ise 0 kabul edilir)
    SET new_fee = IFNULL(:NEW.DETAIL_DAYSLATE * :NEW.DETAIL_LATEFEE, 0);

    -- Gecikme ücretindeki değişiklik
    SET fee_diff = new_fee - old_fee;

    -- Gecikme ücretindeki değişiklik sıfır değilse, üyelik bakiyesini güncelle
    IF fee_diff <> 0 THEN
        UPDATE MEMBERSHIP
        SET MEMBERSHIP_BALANCE = MEMBERSHIP_BALANCE - fee_diff
        WHERE MEMBERSHIP_ID = :NEW.MEMBERSHIP_ID;
    END IF;
```

60. RENTAL tablosundaki kiralama numaraları için otomatik artırmayı etkinleştirin. Değerleri 1100 ile başlatın

```
ALTER TABLE RENTAL
MODIFY COLUMN RENTAL_ID INT AUTO_INCREMENT START WITH 1100;
```

- RENTAL_ID: Bu, kiralama numarasının bulunduğu sütun olmalıdır.
- AUTO_INCREMENT: Bu, her yeni satır eklendiğinde otomatik olarak artacak şekilde ayarlanır.
- START WITH 1100: Bu, otomatik artırmanın ilk değeri olarak 1100'ü ayarlar.

Eğer RENTAL_ID sütunu zaten AUTO_INCREMENT olarak tanımlanmışsa, yalnızca başlangıç değerini değiştiren kısmı kullanmanız yeterlidir.

61. RENTAL tablosuna yeni satırlar eklemek için ***prc_new_rental*** adında bir saklı yordam oluşturun. Yordam aşağıdaki
koşulları sağlamalıdır:
• Üyelik numarası bir parametre olarak sağlanacaktır.
• Üyelik numarasının MEMBERSHIP tablosunda mevcut olduğunu doğrulamak için Count() fonksiyonunu kullanın. Eğer
yoksa, üyeliğin mevcut olmadığını ve veritabanına veri yazılmaması gerektiğini belirten bir mesaj görüntülenmelidir.
• Üyelik mevcutsa, üyelik alın ve bakiye tutarının önceki bakiye olduğunu belirten bir mesaj görüntüleyin. (Örneğin,
üyeliğin 5,00 $ bakiyesi varsa, "Önceki : 5,00 $" mesajını görüntüleyin).
• RENT_DATE değeri için geçerli sistem tarihini ve MEM_NUM için değer olarak sağlanan üyelik numarasını kullanarak kiralama tablosuna yeni bir satır ekleyin. Kira numarası, önceki problemde etkinleştirilen otomatik artıştan
otomatik olarak sağlanmalıdır.

```
DELIMITER $$

CREATE PROCEDURE prc_new_rental(IN membership_num INT)
BEGIN
    DECLARE membership_balance DECIMAL(10, 2);

    -- Üyeliğin mevcut olup olmadığını kontrol et
    IF (SELECT COUNT(*) FROM MEMBERSHIP WHERE MEMBERSHIP_ID = membership_num) = 0 THEN
        -- Üyelik mevcut değilse, mesaj göster
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Üyelik mevcut değil!';
    ELSE
        -- Üyelik mevcutsa, bakiye bilgisini al
        SELECT MEMBERSHIP_BALANCE INTO membership_balance
        FROM MEMBERSHIP
        WHERE MEMBERSHIP_ID = membership_num;

        -- Önceki bakiye mesajı göster
        SELECT CONCAT('Önceki : ', membership_balance) AS Message;

        -- Kiralama kaydını ekle
        INSERT INTO RENTAL (MEMBERSHIP_ID, RENT_DATE)
        VALUES (membership_num, CURDATE());
    END IF;
END $$

DELIMITER ;
```

- **Üyelik kontrolü:** `COUNT()` fonksiyonu ile `MEMBERSHIP` tablosunda üyelik numarasının mevcut olup olmadığını kontrol eder.
- **Bakiye bilgisi:** Eğer üyelik mevcutsa, üyeliğin bakiyesi alınır ve önceki bakiye mesajı görüntülenir.
- **Kiralama kaydı:** Eğer üyelik geçerliyse, `RENTAL` tablosuna geçerli sistem tarihi (`CURDATE()`) ile yeni bir kiralama kaydı eklenir.

62. DETAILRENTAL tablosuna yeni satırlar eklemek için ***prc_new_detail*** adında bir saklı yordam oluşturun. Yordam
aşağıdaki gereksinimleri karşılamalıdır:

- Video numarası bir parametre olarak sağlanacaktır.
- Video numarasının VIDEO tablosunda mevcut olduğunu doğrulayın. Eğer yoksa, videonun mevcut olmadığını belirten bir mesaj görüntüleyin ve veritabanına herhangi bir veri yazmayın.
- Video numarası mevcutsa, video için VID_STATUS değerinin "IN" olduğunu doğrulayın. Durum "IN" değilse, tekrar
kiralanabilmesi için videonun iadesinin girilmesi gerektiğini belirten bir mesaj görüntüleyin ve veritabanına herhangi bir
veri yazmayın.
- Durum "IN" ise, videonun PRICE_RENTFEE, PRICE_DAILYLATEFEE ve PRICE_RENTDAYS değerlerini PRICE tablosundan alın.
- PRICE_RENTDAYS içindeki gün sayısını geçerli sistem tarihindeki 11:59:59PM (saat:dakika:saniye) değerine ekleyerek video kiralama için son tarihi hesaplayın.
- RENT_NUM_SEQ tarafından döndürülen önceki değeri RENT_NUM olarak, parametrede sağlanan video numarasını VID_NUM olarak, PRICE_RENTFEE değerini DETAIL_FEE değeri olarak, yukarıda DETAIL_DUEDATE için hesaplanan son tarihi, PRICE_DAILYLATEFEE değerini DETAIL_DAILYLATEFEE değeri olarak ve null değerini DETAIL_RETURNDATE için kullanarak DETAILRENTAL tablosuna yeni bir satır
ekleyin.

```
DELIMITER $$



CREATE PROCEDURE prc_new_detail(IN video_num INT)

BEGIN

   DECLARE vid_status VARCHAR(4);

   DECLARE rent_fee DECIMAL(10, 2), daily_late_fee DECIMAL(10, 2);

   DECLARE rent_days INT;

   DECLARE due_date DATETIME;



   -- Video ve durum kontrolü

   IF (SELECT VID_STATUS FROM VIDEO WHERE VID_NUM = video_num) != 'IN' THEN

      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Video mevcut değil veya iade edilmelidir!';

   ELSE

      -- Fiyat ve süre bilgilerini al

      SELECT PRICE_RENTFEE, PRICE_DAILYLATEFEE, PRICE_RENTDAYS INTO rent_fee, daily_late_fee, rent_days
```

```
    FROM PRICE WHERE VID_NUM = video_num;


    -- Son teslim tarihi hesapla

    SET due_date = CONCAT(DATE_ADD(CURDATE(), INTERVAL rent_days
DAY), ' 23:59:59');


    -- Yeni satır ekle

    INSERT INTO DETAILRENTAL (RENT_NUM, VID_NUM, DETAIL_FEE,
DETAIL_DUEDATE, DETAIL_DAILYLATEFEE)

    VALUES (NEXTVAL('RENT_NUM_SEQ'), video_num, rent_fee, due_date,
daily_late_fee);

  END IF;

END $$


DELIMITER ;
```

63. Kiralanan videoların iadesi ile ilgili verileri girmek için ***prc_return_video*** adında bir saklı yordam oluşturun. Yordam
aşağıdaki gereksinimleri karşılamalıdır.

- Video numarası bir parametre olarak sağlanacaktır.
- Video numarasının VIDEO tablosunda mevcut olduğunu doğrulayın. Mevcut değilse, sağlanan video numarasının
bulunamadığını belirten bir mesaj görüntüleyin ve veritabanına herhangi bir veri yazmayın.
- Video numarası mevcutsa, videonun DETAIL- RENTAL içinde iade tarihi olmayan yalnızca bir kaydı olduğundan
emin olmak için bir Count() işlevi kullanın. DETAILRENTAL'de birden fazla satır videonun kiralandığını ancak iade
edilmediğini gösteriyorsa, videonun birden fazla bekleyen kiralaması olduğuna dair bir hata mesajı görüntüleyin ve
veritabanına herhangi bir veri yazmayın.
- Videonun ödenmemiş kiralaması yoksa VİDEO tablosunda video durumunu "VAR" olarak güncelleyin ve videonun
ödenmemiş kiralaması olmadığını ancak artık kiralanabileceğini belirten bir mesaj görüntüleyin. Videonun yalnızca bir
bekleyen kiralaması varsa, iade tarihini geçerli sistem tarihine güncelleyin ve VİDEO tablosunda o video için video
durumunu "GİRİŞ" ogüncelleyin. Ardından videonun başarıyla iade edildiğini belirten bir mesaj görüntüleyin.