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.

1. 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.

1. "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.

1. 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.

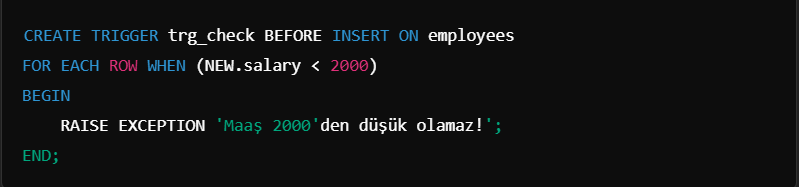
INSERT + Alt Sorgu → Mevcut tabloya veri ekler.

9. Dizi nedir? Sözdizimini yazınız.

Sequence (Dizi), otomatik artan değerler üretir.

1. Tetikleyici nedir ve amacı nedir? Bir örnek veriniz.

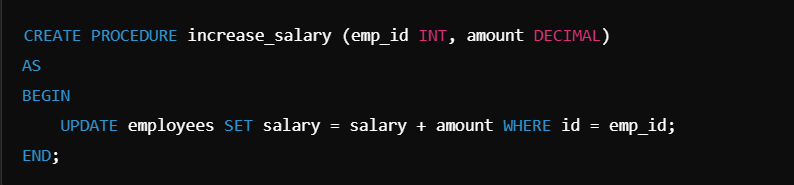
Trigger (Tetikleyici), belirli bir işlem (INSERT, UPDATE, DELETE) gerçekleştiğinde otomatik olarak çalışır.



11. Saklı yordam nedir ve neden özellikle yararlıdır? Bir

örnek verin.

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



Çağırmak için:



PROBLEMLER

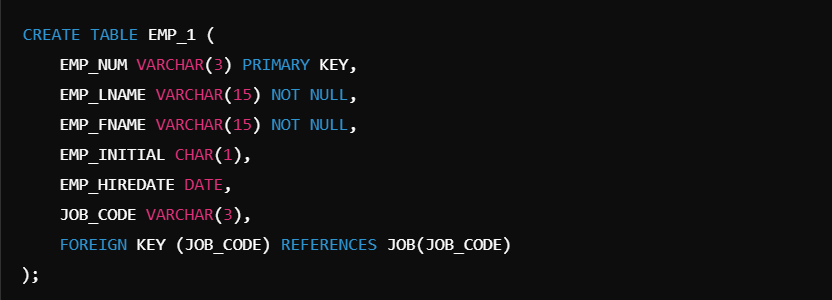
1. EMP\_1 adlı bir tablo için yalnızca tablo yapısını oluşturacak SQL kodunu yazın. Bu tablo EMPLOYEE tablosunun bir alt

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

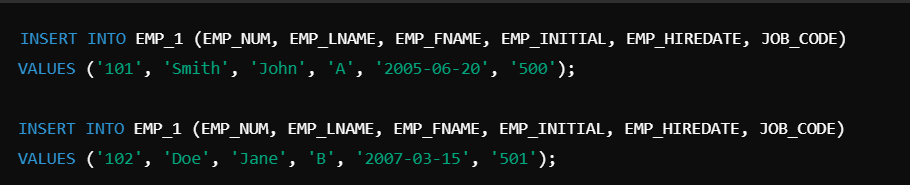
ayrıca EMP\_LNAME ve EMP\_FNAME içinde null girişleri önlemelidir.

EMP\_1 Tablosunun Yapısını Oluşturma



1. 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

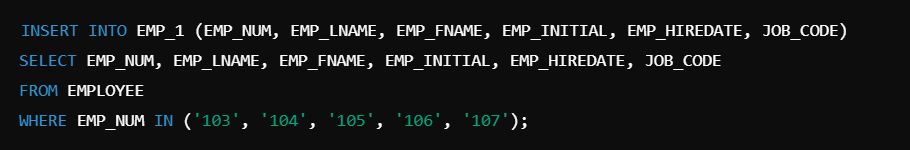


3. Zaten var olan EMPLOYEE tablosunu kullanarak, EMPLOYEE tablosundan kalan satırları EMP\_1 tablosuna eklemek

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



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:



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

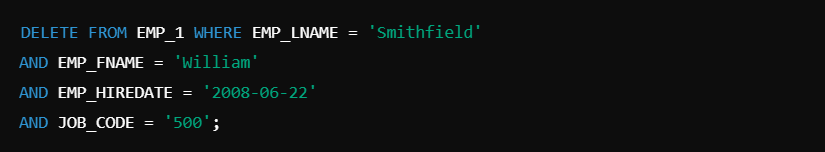


6. İş kodu 22 Haziran 2008'de işe alınan William Smithfield'ın satırını silmek için SQL kodunu yazın

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



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



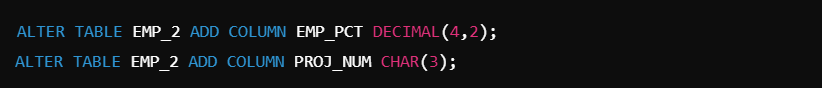
8. EMP\_2 tablosunu kullanarak, EMP\_PCT ve PROJ\_NUM niteliklerini EMP\_2'ye ekleyecek SQL kodunu yazın.

EMP\_PCT, her çalışana ödenecek ikramiye yüzdesidir. Yeni öznitelik özellikleri şunlardır:

EMP\_PCT

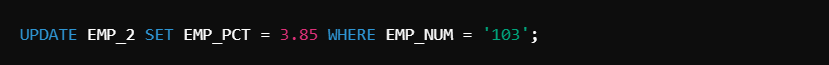
DECIMAL(4,2)

PROJ\_NUM CHAR(3)



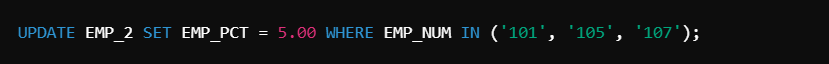
9. EMP\_2 tablosunu kullanarak, çalışan numarası (EMP\_NUM) 103 olan kişinin EMP\_PCT değerini 3,85 olarak

değiştirmek için SQL kodunu yazın.



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.



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.



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).



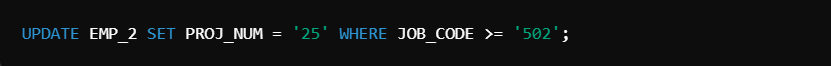
13. EMP\_2 tablosunda tek bir komut dizisi kullanarak, iş sınıflandırması (JOB\_CODE) 500 olan tüm çalışanlar için proje

numarasını (PROJ\_NUM) 18 olarak değiştirecek SQL kodunu yazın.



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.

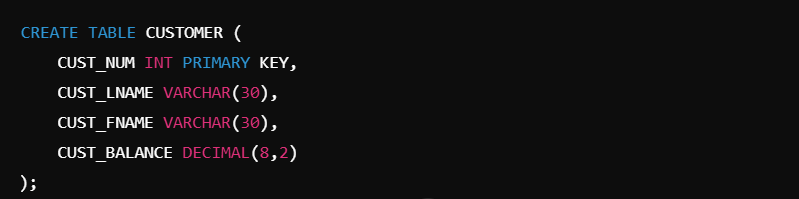


15. PROJ\_NUM'u 1 Ocak 1998'den önce işe alınmış ve iş kodu en az 501 olan çalışanlar için 14 olarak değiştirecek SQL kodunu

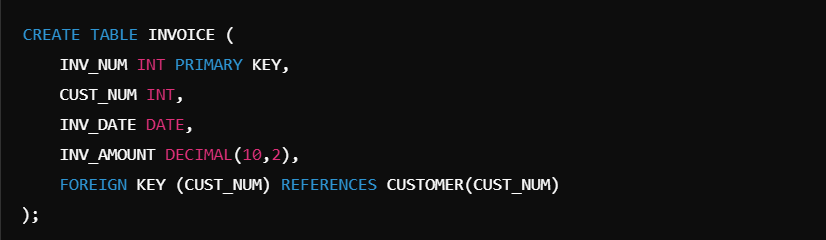
yazın. Problem 7-15'i bitirdiğinizde, EMP\_2 tablosu Şekil P8.15'te gösterilen verileri içerecektir.



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.

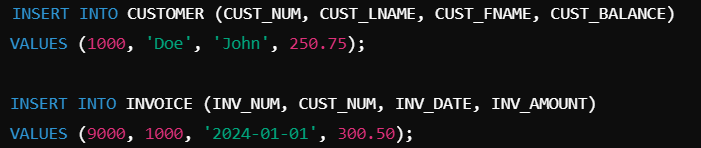


17. Şekil P8.16'da gösterilen INVOICE tablo yapısını oluşturun. Fatura numarası tamsayı değerlerini saklamalıdır. Fatura tarihi, tarih değerlerini saklamalıdır. Fatura tutarı, ondalık basamağın solunda 8 kadar ve ondalık basamağın sağında iki haneye kadar desteklemelidir.



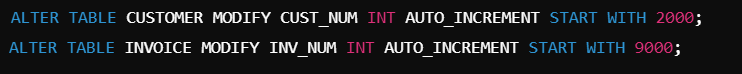
18. Verileri Şekil P8.16'da gösterildiği gibi Problem 16'da oluşturduğunuz CUSTOMER tablosuna eklemek için gerekli SQL komutları kümesini yazın.

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



20. Müşteri numaraları için değerler oluşturmak üzere otomatik artırmayı etkinleştirin. Değerler 2000 ile başlamalıdır.

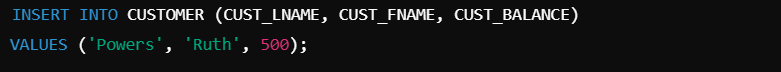
21. Fatura numaraları için değerler oluşturmak üzere otomatik artırmayı etkinleştirin. Değerler 9000 değeri ile başlamalıdır.



22. Müşteri numarasını otomatik olarak oluşturmak için Problem 20'deki otomatik artırmayı kullanarak aşağıdaki müşteriyi

CUSTOMER ekleyin:

'Powers', 'Ruth', 500

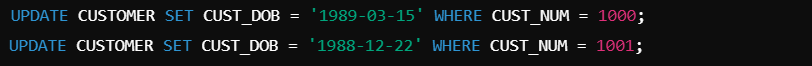


23. CUSTOMER tablosunu, tarih verilerini saklaması gereken müşterinin doğum tarihini (CUST\_DOB) içerecek şekilde değiştirin



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.



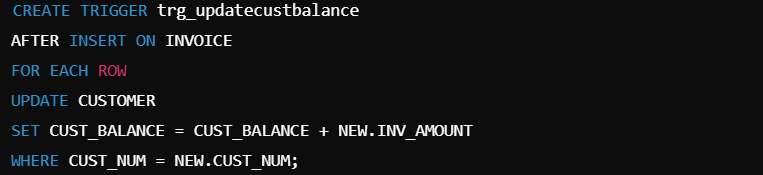
26. Yeni bir fatura kaydı girildiğinde CUSTOMER tablosundaki CUST\_BALANCE değerini güncellemek için

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

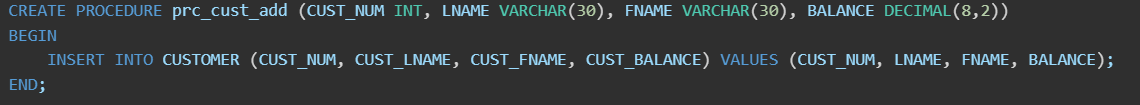


27. CUSTOMER tablosuna yeni bir müşteri eklemek için **prc\_cust\_add** adında bir prosedür yazın. Yeni kayıtta aşağıdaki

değerleri kullanın:

1002, 'Rauthor', 'Peter', 0.00

(Kodunuzun doğru olduğundan emin olmak için prosedürü çalıştırmalı ve yeni müşterinin eklendiğini doğrulamalısınız).

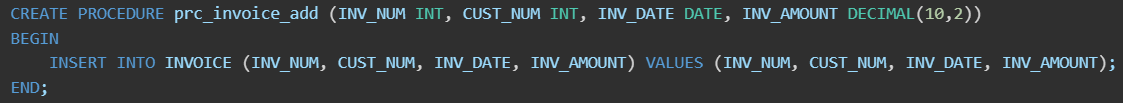


28. INVOICE tablosuna yeni bir fatura kaydı eklemek için **prc\_invoice\_add** adında bir prosedür yazın. Yeni kayıtta

aşağıdaki değerleri kullanın:

8006, 1000, '2022-04-30', 301.72

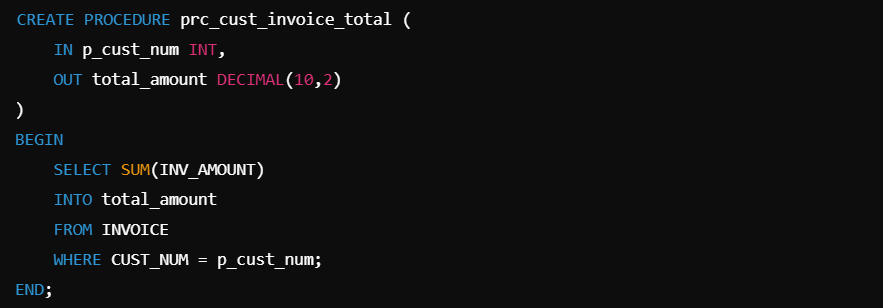
(Kodunuzun doğru olduğundan emin olmak için prosedürü çalıştırmalı ve yeni faturanın eklendiğini doğrulamalısınız).



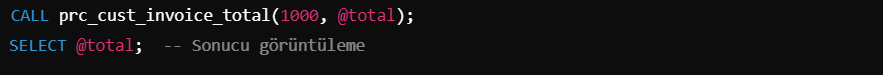
29. Bir fatura silindiğinde müşteri bakiyesini güncellemek için bir tetikleyici yazın. Tetikleyiciye **trg\_updatecustbalance2** adını verin.

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.



Kullanımı:

Örneğin, CUST\_NUM = 1000 olan müşterinin toplam fatura tutarını almak için :

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;

1. 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

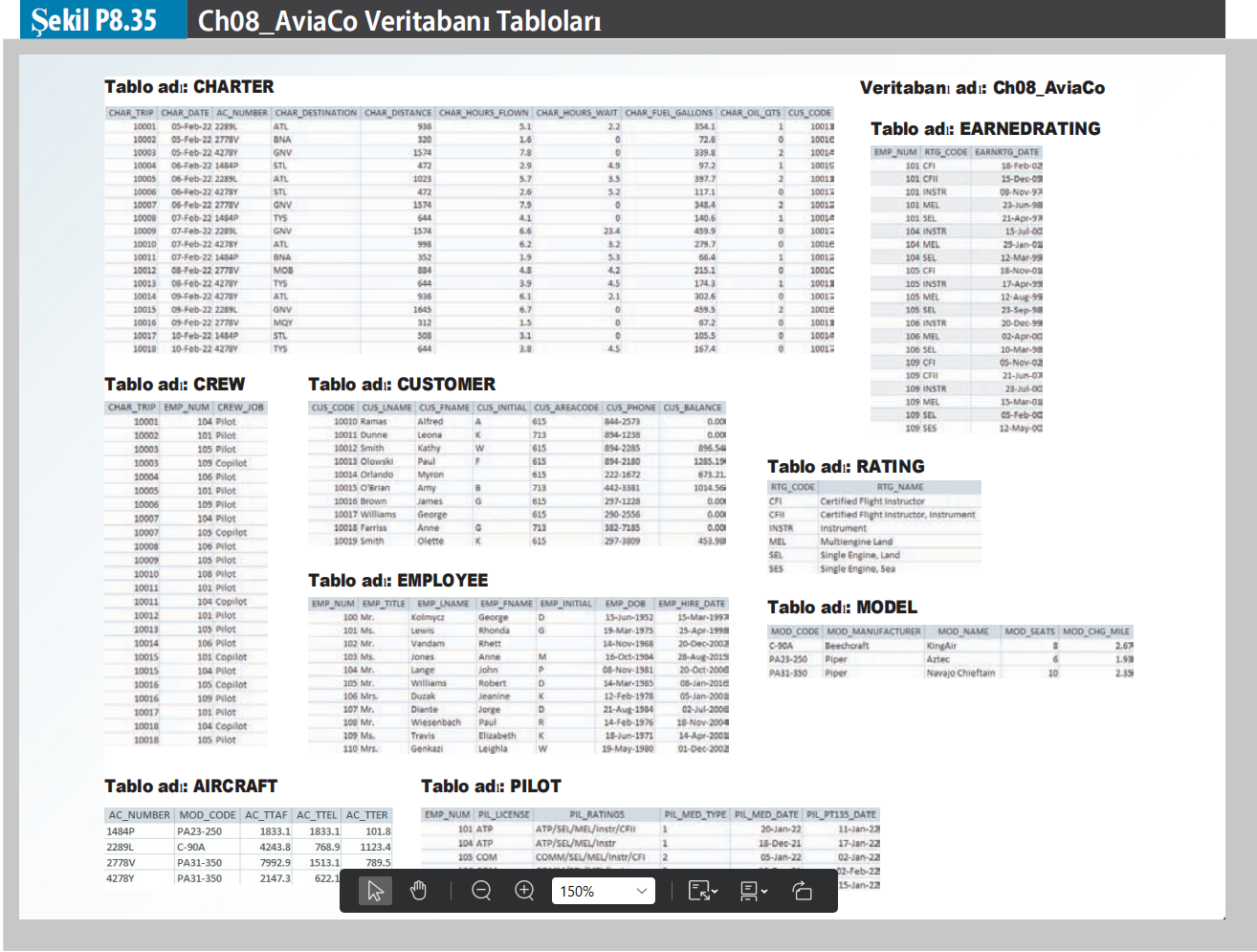
UPDATE CUSTOMER

SET BALANCE = BALANCE + TOTINV

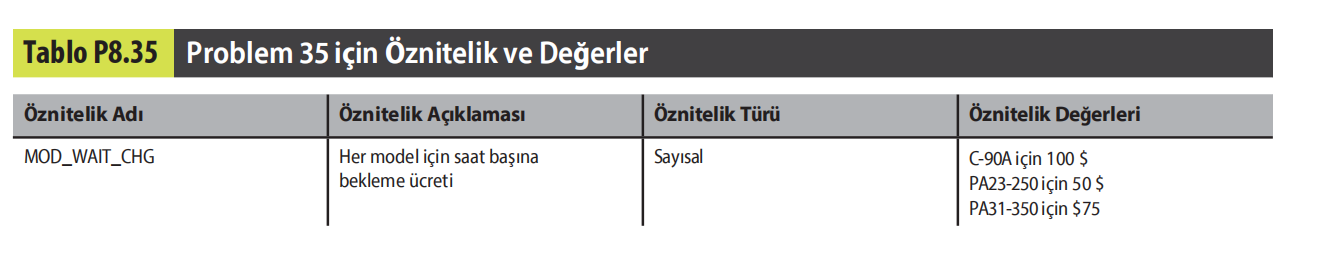
WHERE CUSTOMER\_ID = (SELECT CUSTOMER\_ID FROM INV WHERE INVOICE\_ID = invoice\_id);

END;

Problem 35-46 üzerinde çalışmak için Şekil P8.35'te gösterilen Ch08\_AviaCo veritabanını kullanın.



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



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.



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;

1. 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;

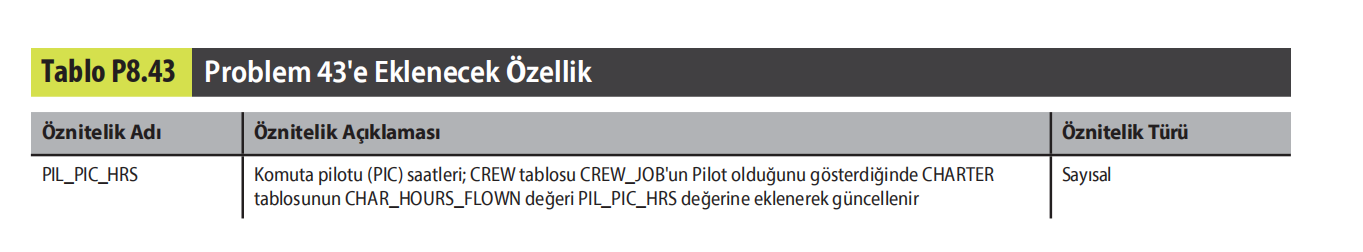
UPDATE CHARTER

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.



ALTER TABLE PILOT

ADD PIL\_PIC\_HRS NUMERIC;

ALTER TABLE PILOT

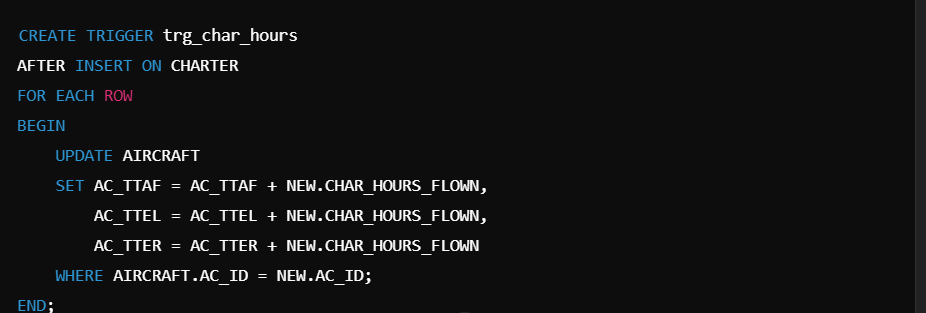
ADD PIL\_PIC\_HRS NUMERIC DEFAULT 0;

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:

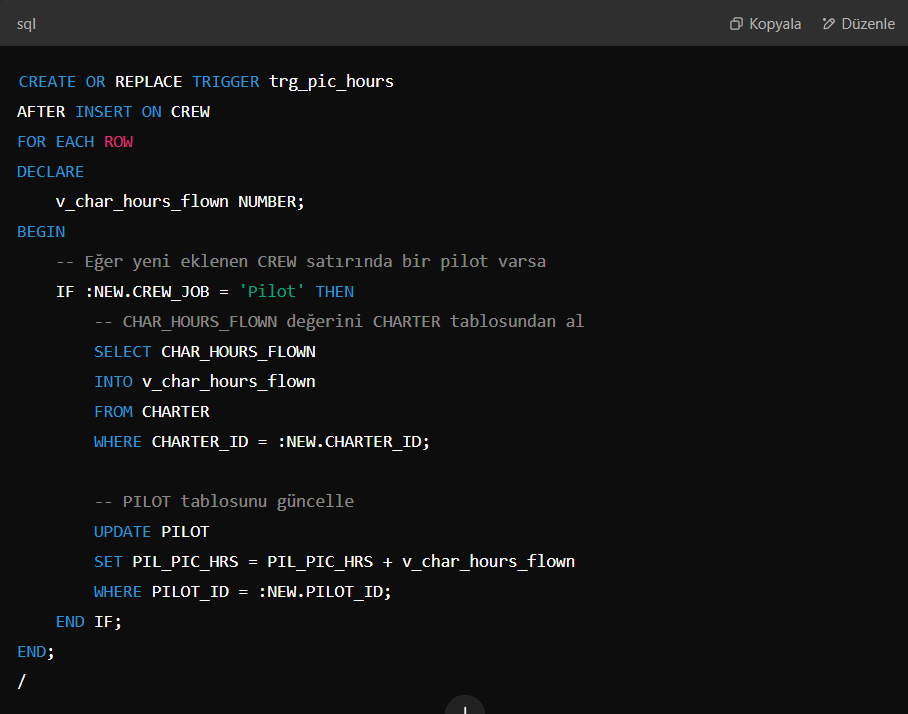


45. Yeni bir CREW satırı eklendiğinde ve CREW tablosu bir Pilot CREW\_JOB girişi kullandığında PILOT tablosunu

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

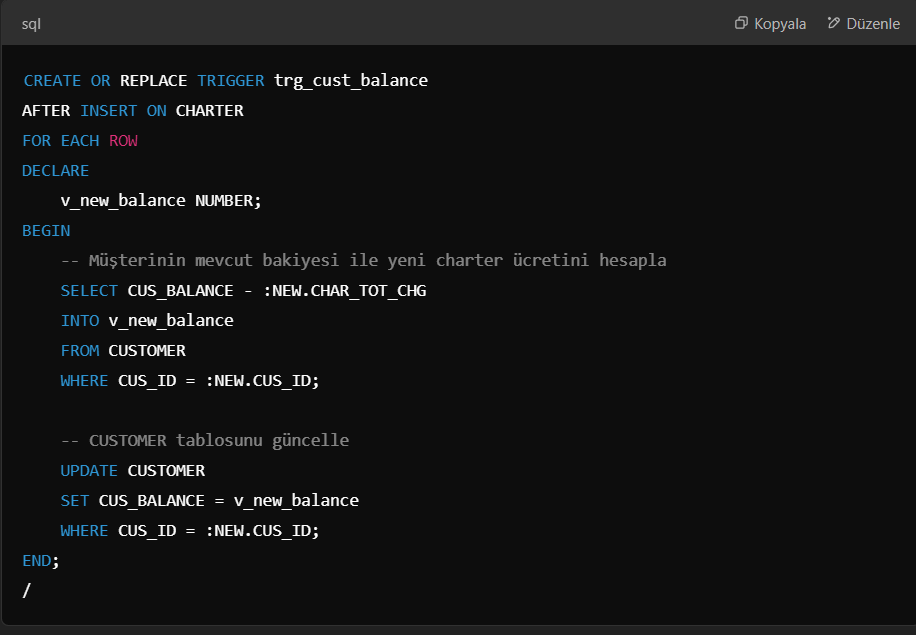
CHAR\_HOURS\_FLOWN değerini kullanın

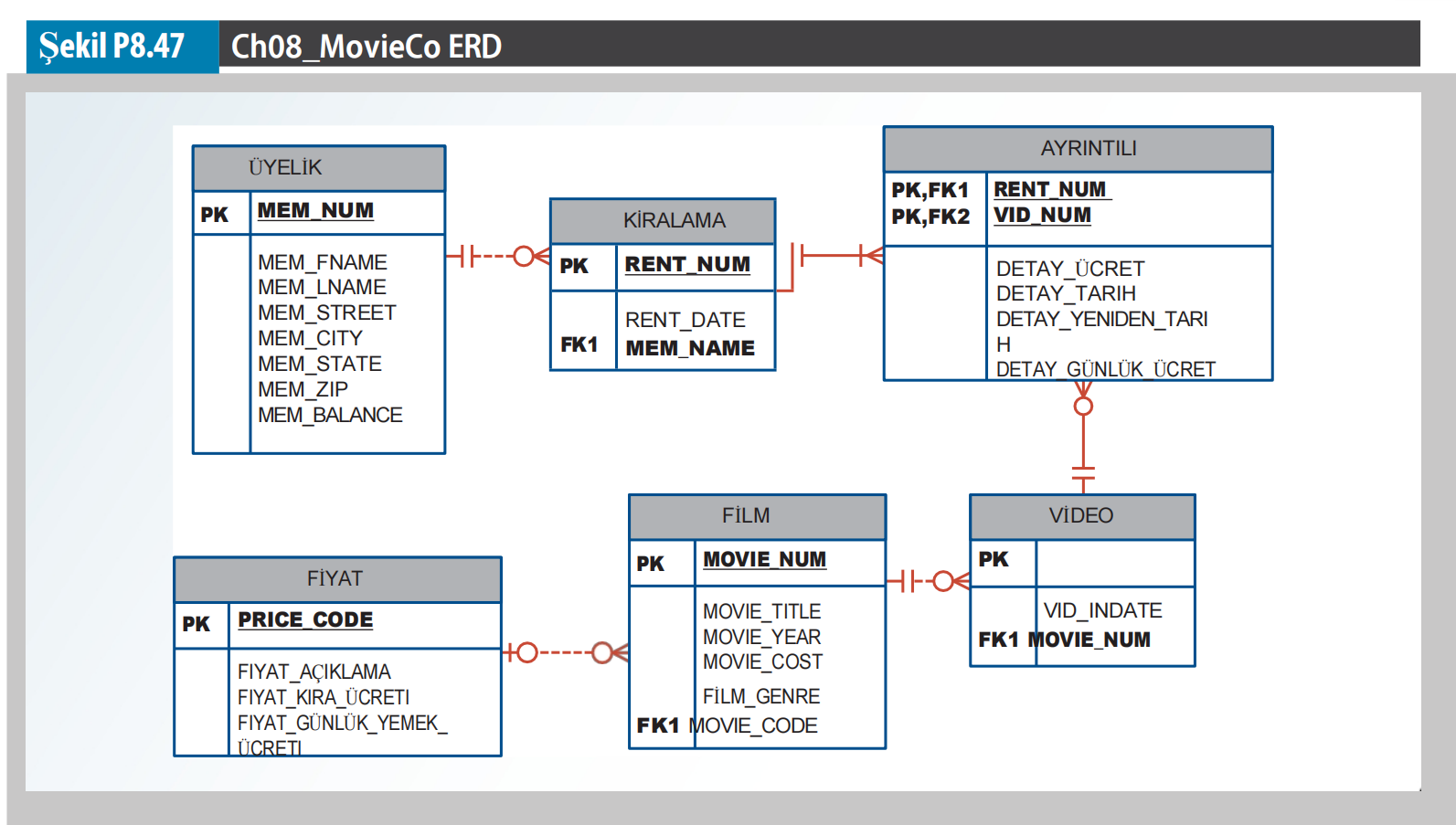


46. Yeni bir CHARTER satırı eklendiğinde CUSTOMER tablosunun CUS\_BALANCE değerini otomatik olarak güncelleyen

**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)





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));

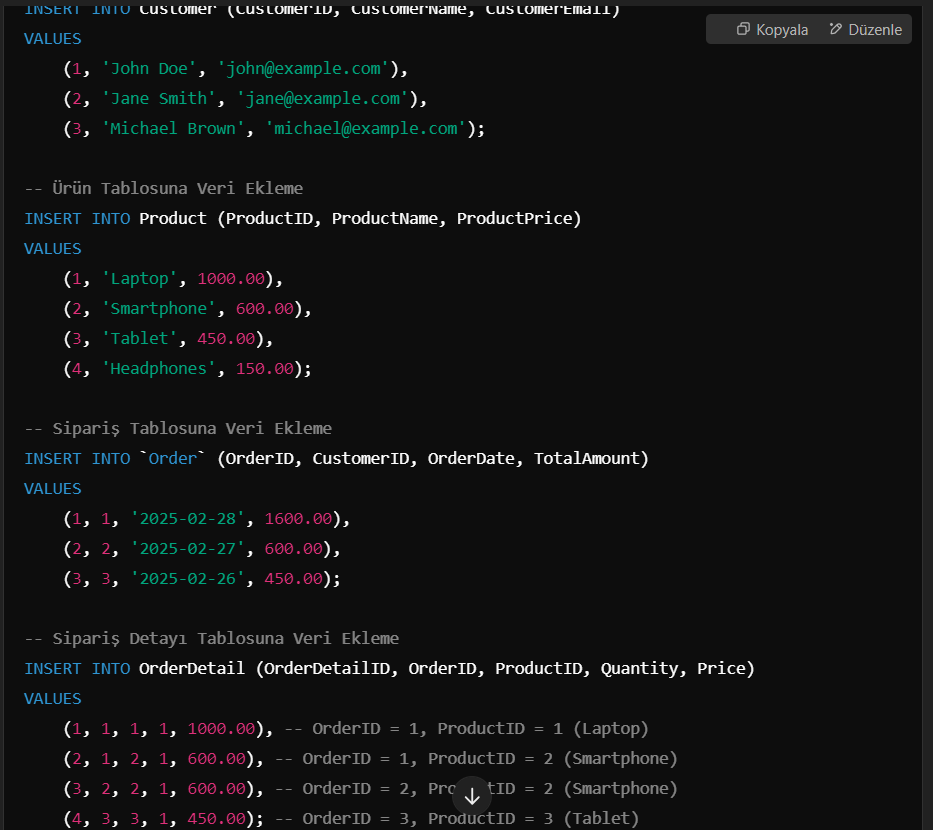
CREATE TABLE `Order` (OrderID INT PRIMARY KEY, CustomerID INT, OrderDate DATE, TotalAmount DECIMAL(10, 2), FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID));

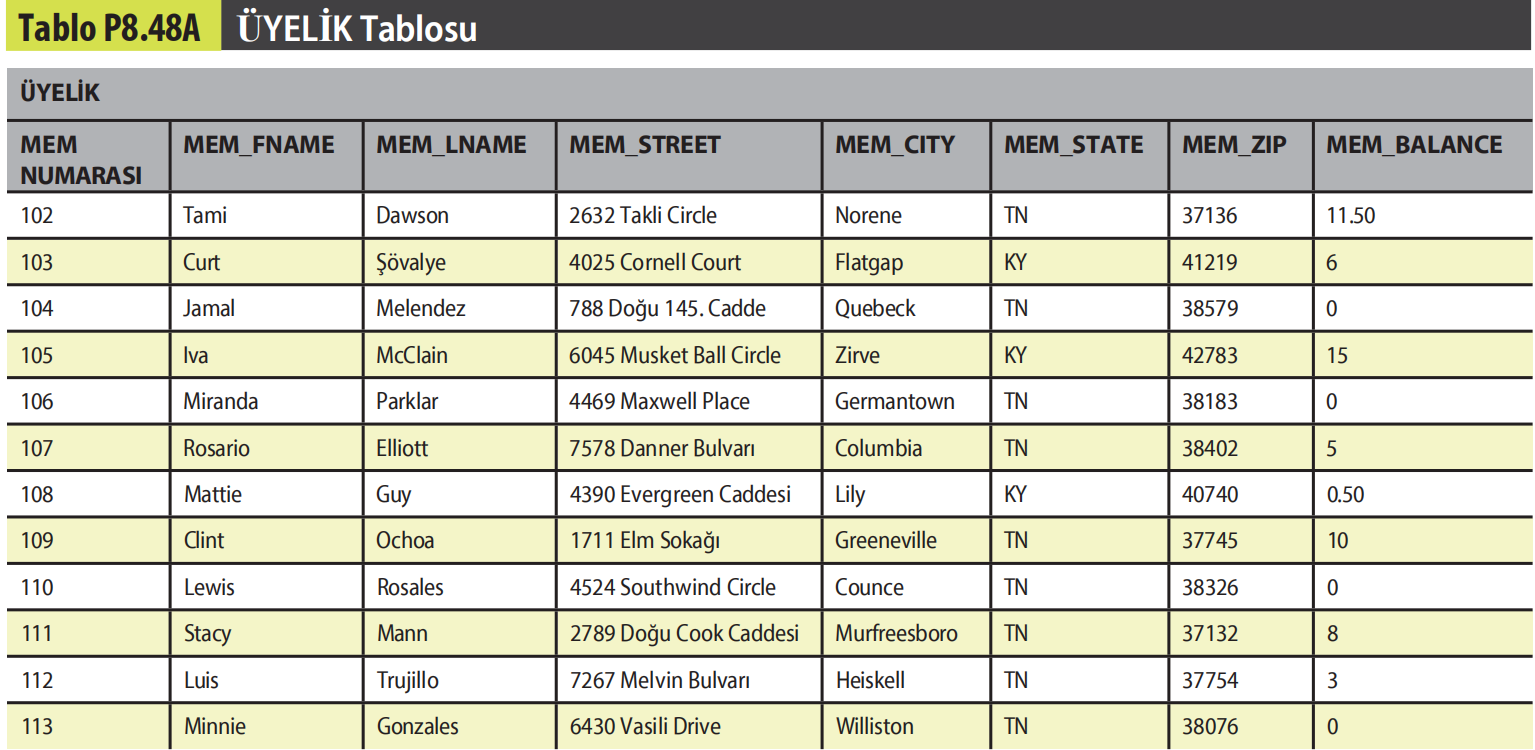
CREATE TABLE OrderDetail (OrderDetailID INT PRIMARY KEY, OrderID INT, ProductID INT, Quantity INT, Price DECIMAL(10, 2), FOREIGN KEY (OrderID) REFERENCES `Order`(OrderID), FOREIGN KEY (ProductID) REFERENCES Product(ProductID));

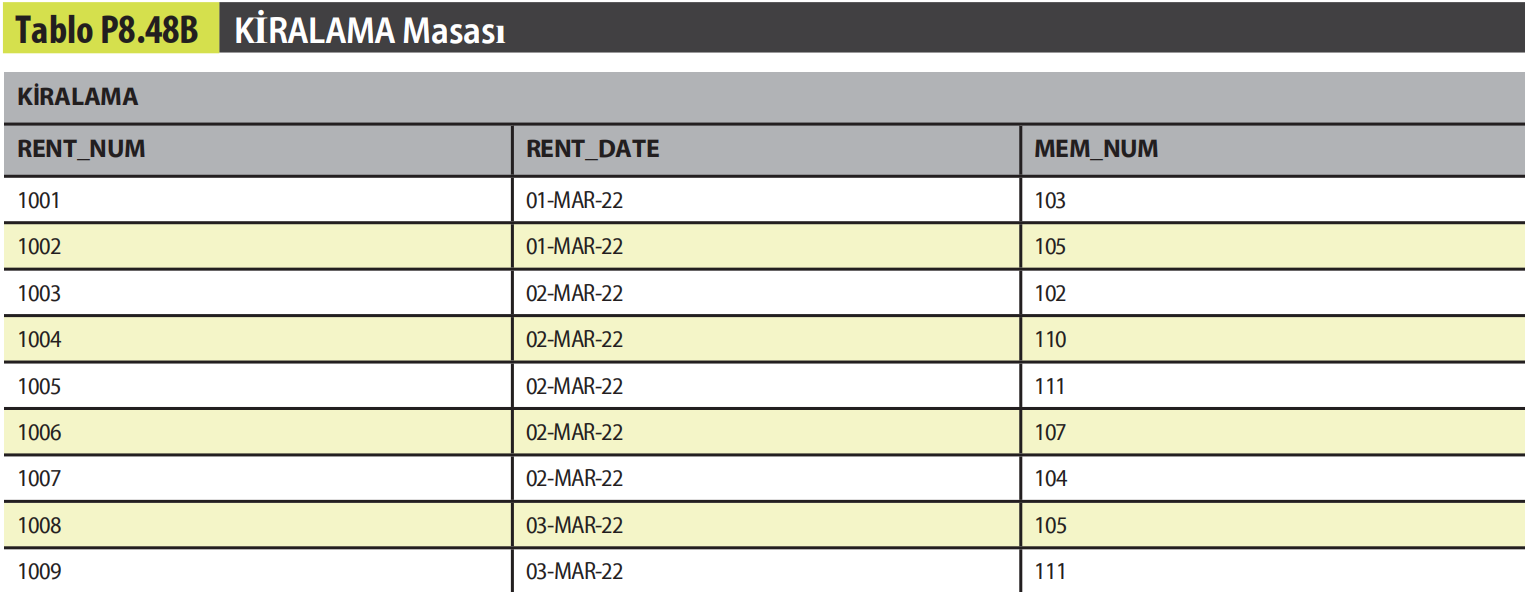
48. Aşağıdaki tablolar, veritabanında tutulacak verilerin çok küçük bir bölümünü sağlamaktadır. Verilerin test amacıyla

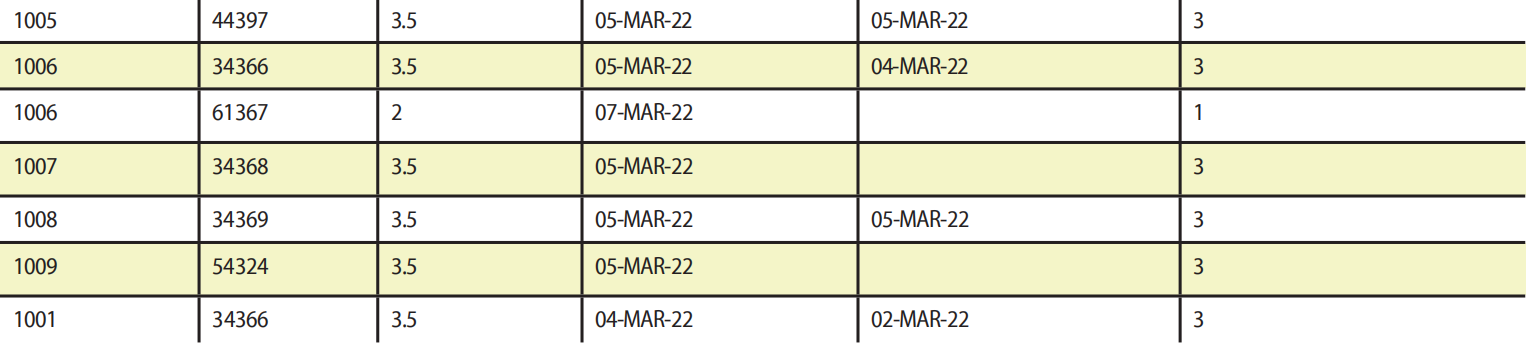
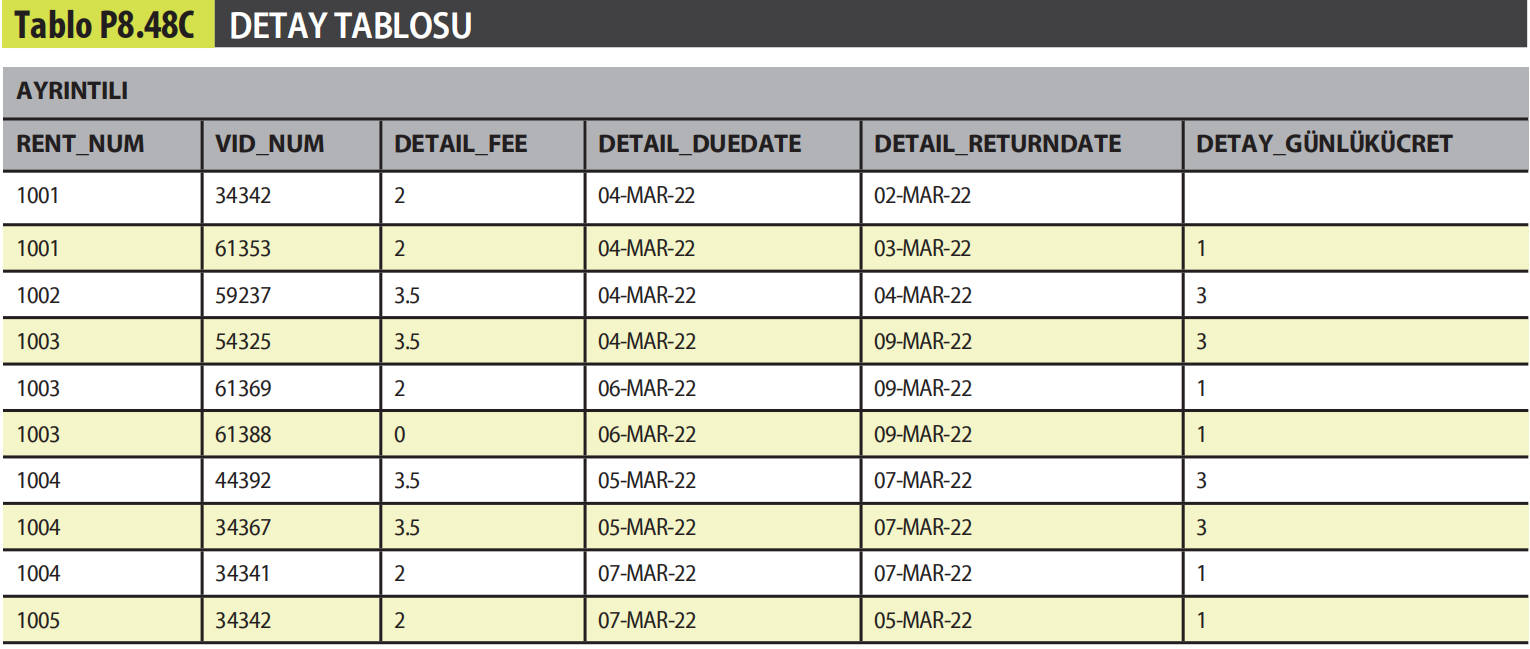
veritabanına eklenmesi gerekmektedir. Aşağıdaki verileri Problem 47'de oluşturulan tablolara yerleştirmek için gerekli

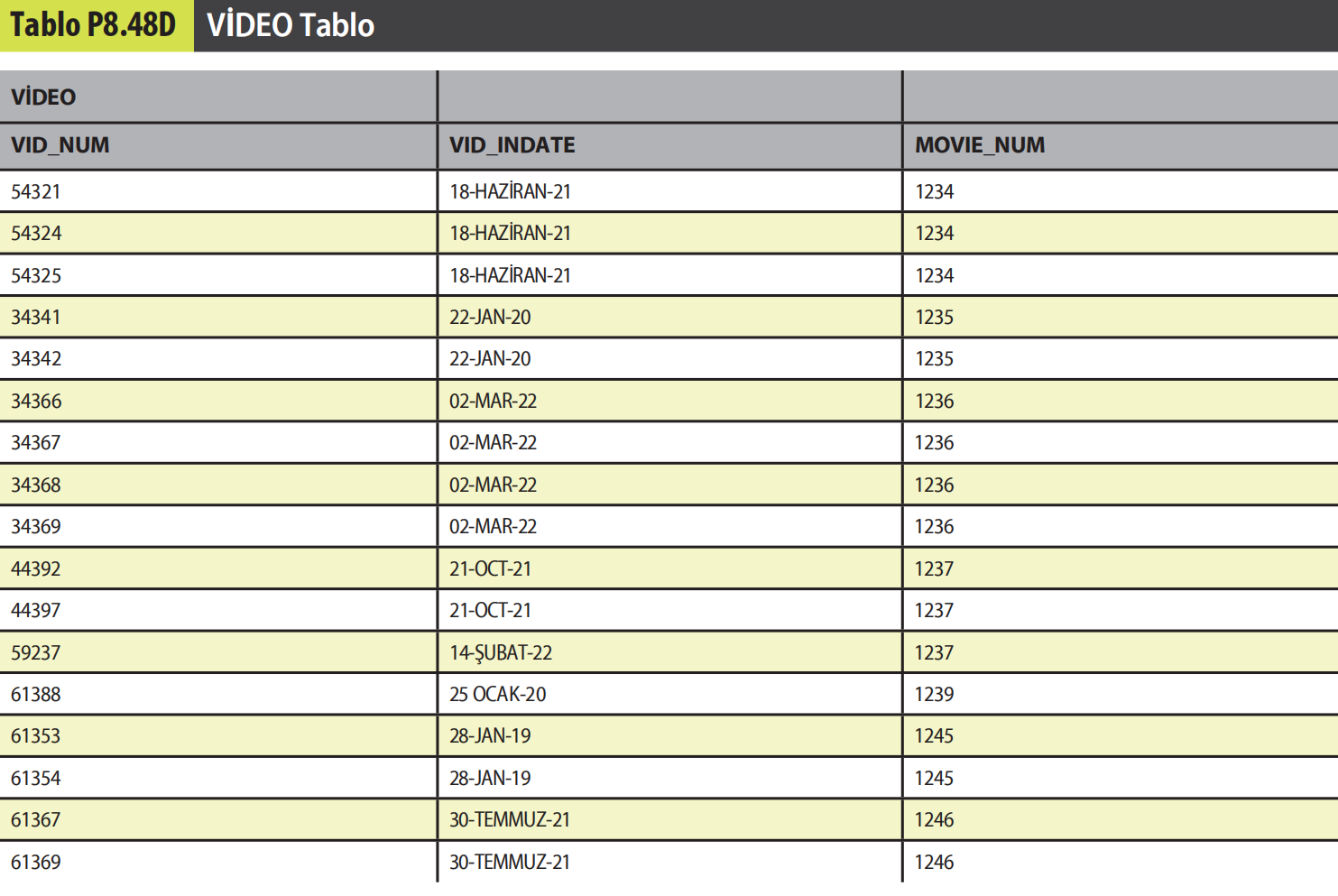
INSERT komutlarını yazın (DBMS'niz gerektiriyorsa, satırları kalıcı olarak kaydettiğinizden emin olun)

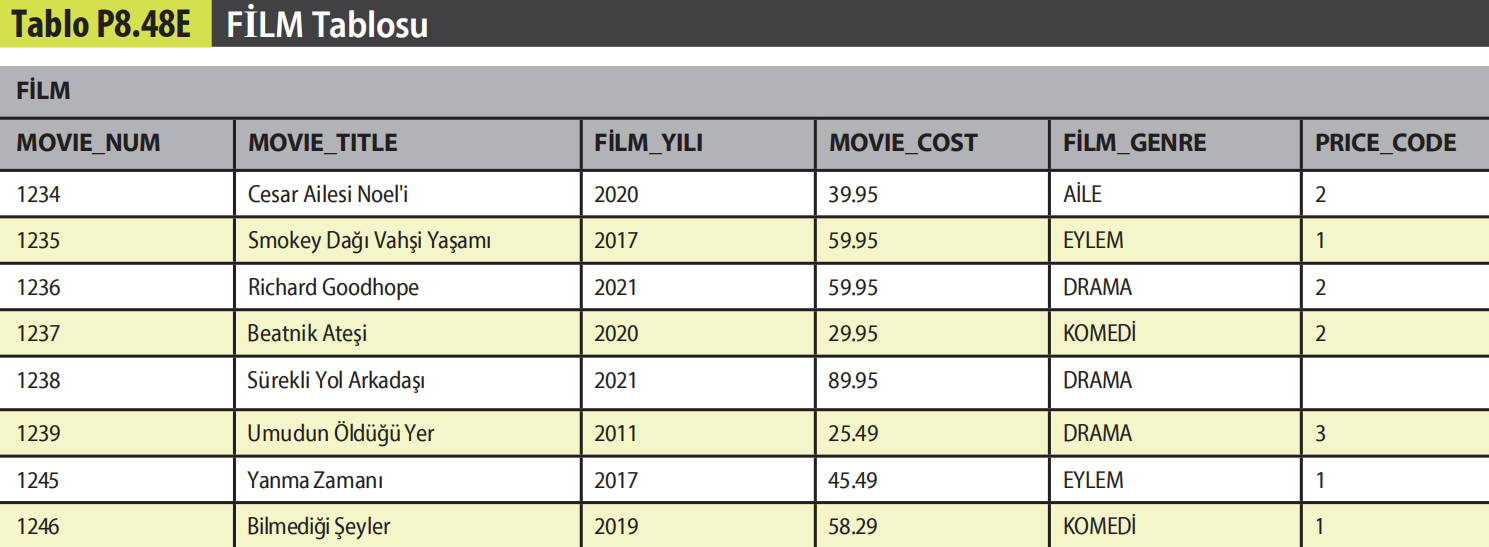


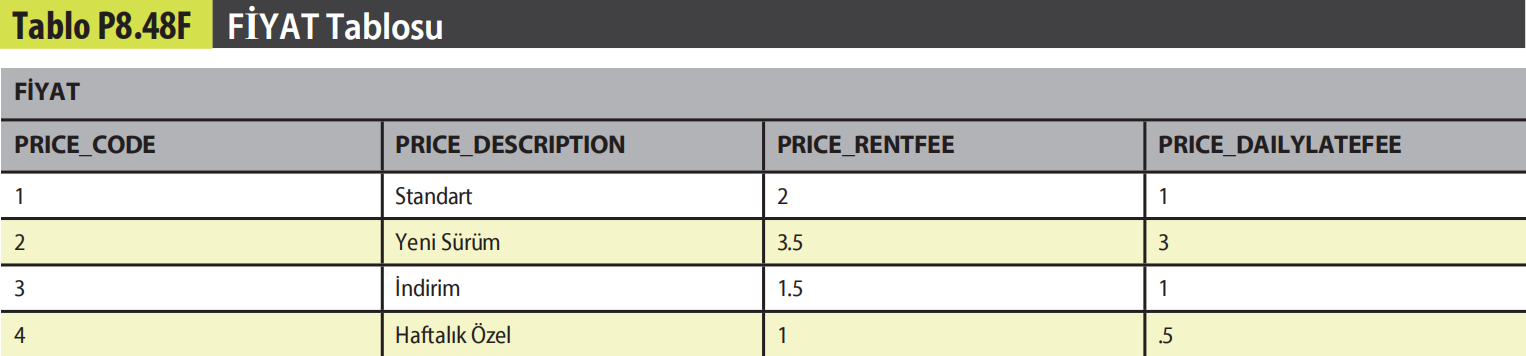












**49-63. Sorular için Problem 47'de oluşturulan tabloları ve Prob- lem 48'de bu tablolara yüklenen verileri kullanın**

1. 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.

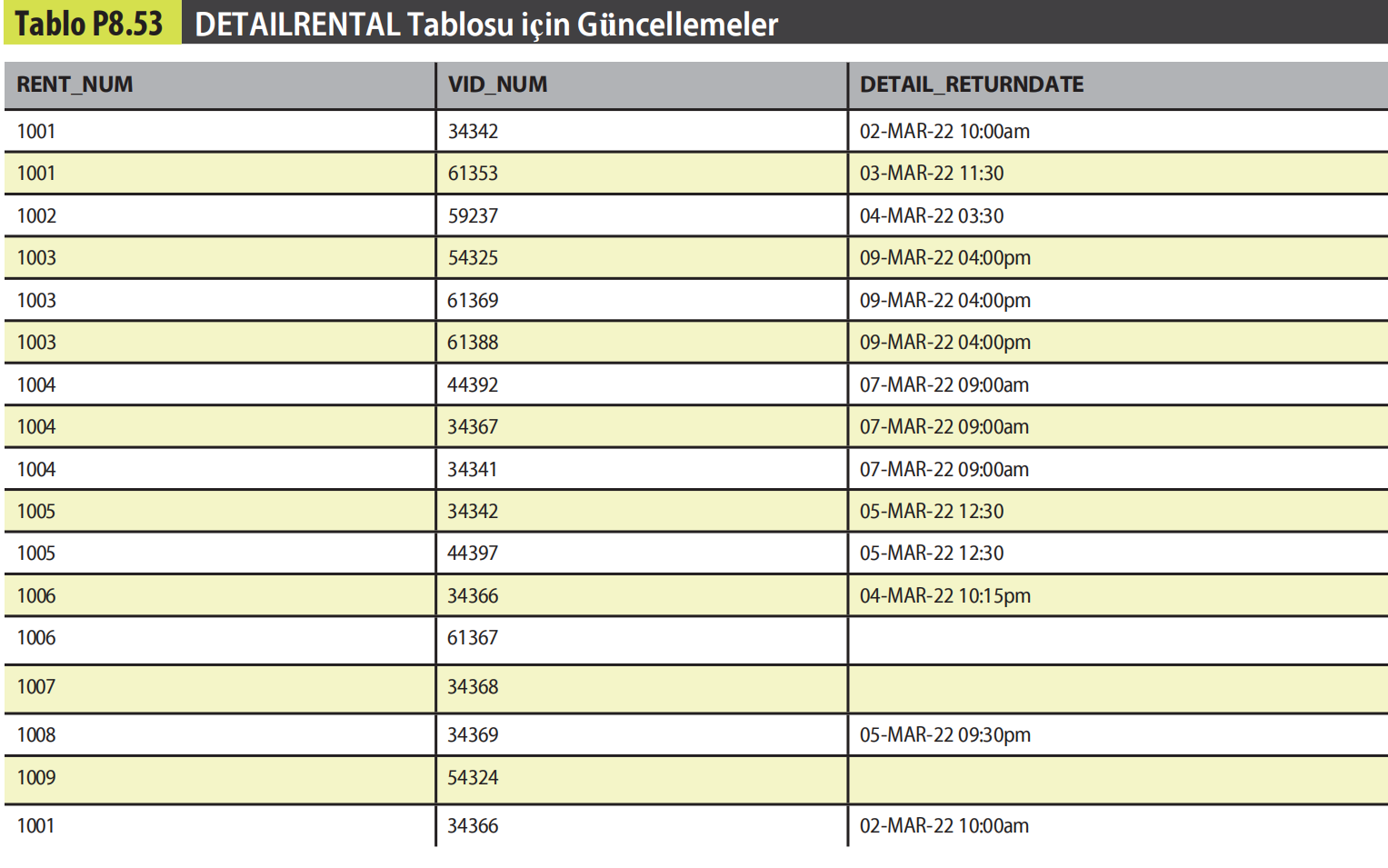
· DETAIL\_DAYSLATE: Yeni eklenen türetilmiş öznitelik adı.

· INT(3): Üç haneli tam sayı. (Burada sayı 0-999 arası bir değeri ifade eder).

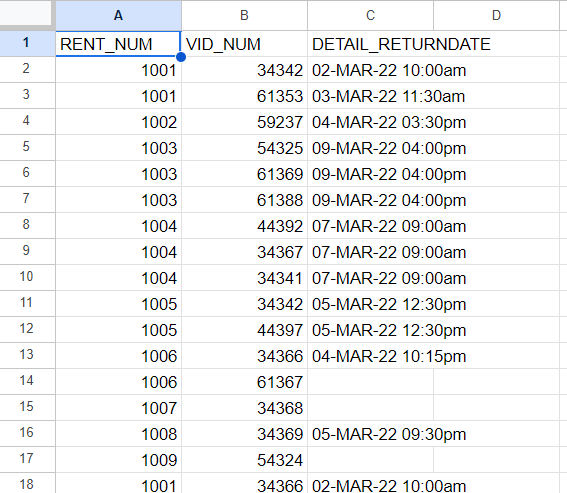
· NULL: Bu öznitelik için NULL değerlerinin kabul edilmesine izin verilir.

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'teki DETAILRENTAL tablosunun güncellenmiş hali aşağıdadır:



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:**

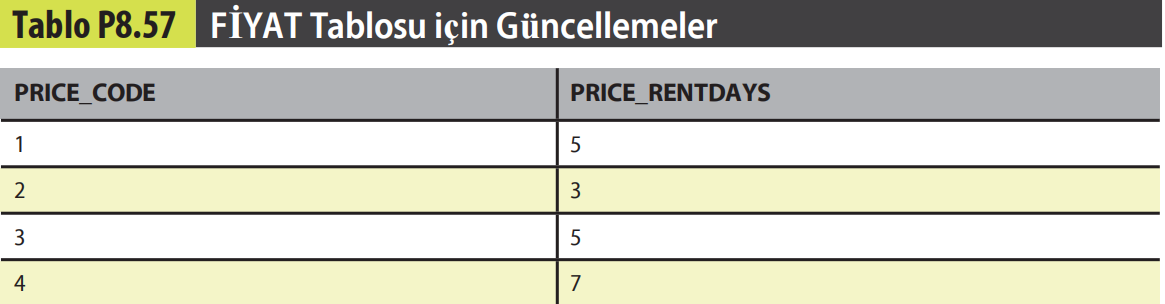
1. ALTER TABLE PRICE: PRICE tablosunu değiştireceğimizi belirtir.
2. ADD PRICE\_RENTDAYS TINYINT: PRICE\_RENTDAYS adında, en fazla 255 (yani iki basamaklı tam sayıları saklayabilecek) bir tamsayı sütunu ekler. TINYINT veri tipi, küçük tamsayı değerlerini saklamak için kullanılır ve genellikle 0 ile 255 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.

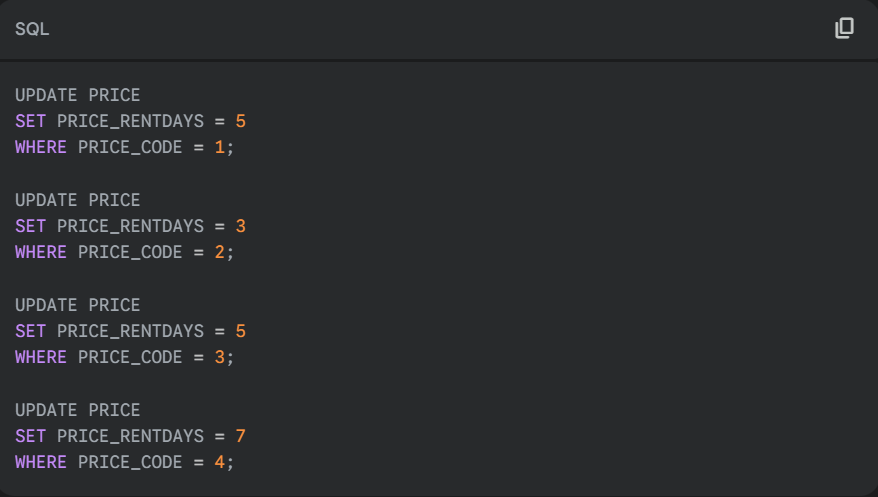
**Bu komutu çalıştırdıktan sonra:**

* 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.





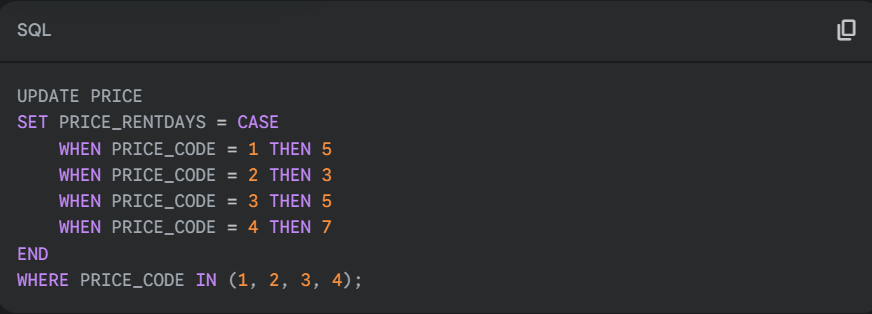
**Her bir sorgunun yaptığı işlem:**

* 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:**



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.

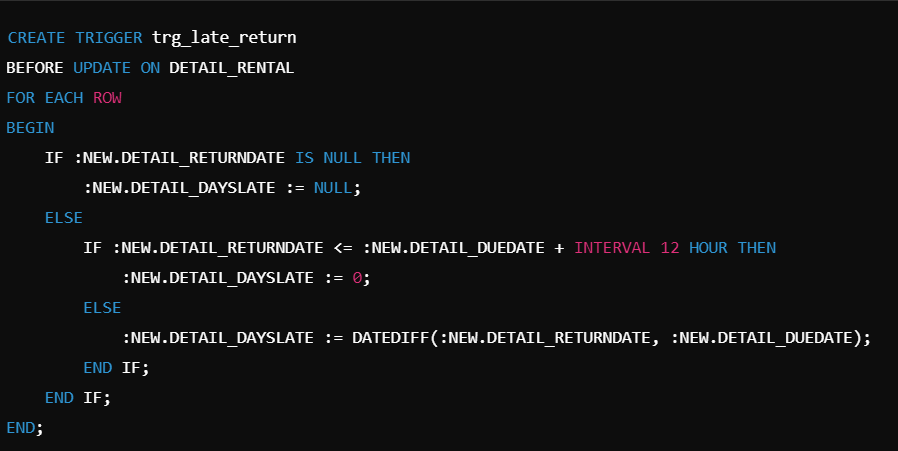
• İade tarihi boş değilse, videonun geç iade edilip edilmediğini geciken günler belirlemelidir.

• 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.



· 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

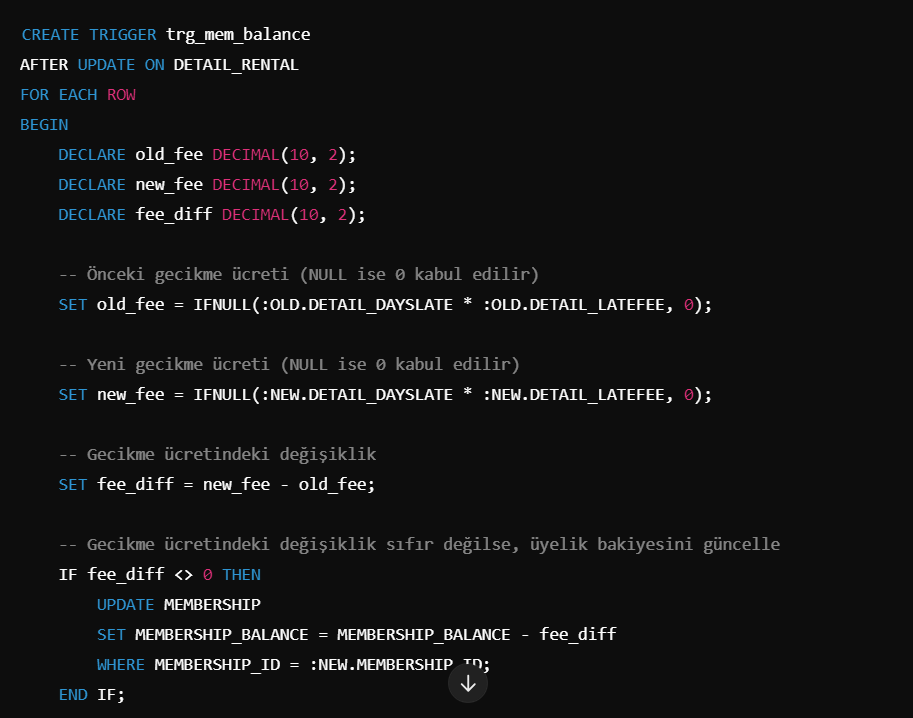
ücretinin değeri artık boşsa, sıfır (0) olarak kabul edin.

• Bu video kiralama gecikme ücretindeki değişikliği belirlemek için gecikme ücretinin önceki değerini gecikme

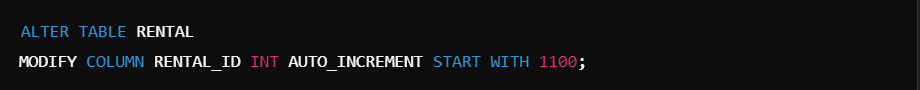
ücretinin mevcut değerinden çıkarın.

• Yukarıda hesaplanan gecikme ücretindeki değişiklik sıfır (0) değilse, üyelik bakiyesini bu kiralamayla ilişkili üyelik

için hesaplanan tutar kadar güncelleyin.



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



* 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.

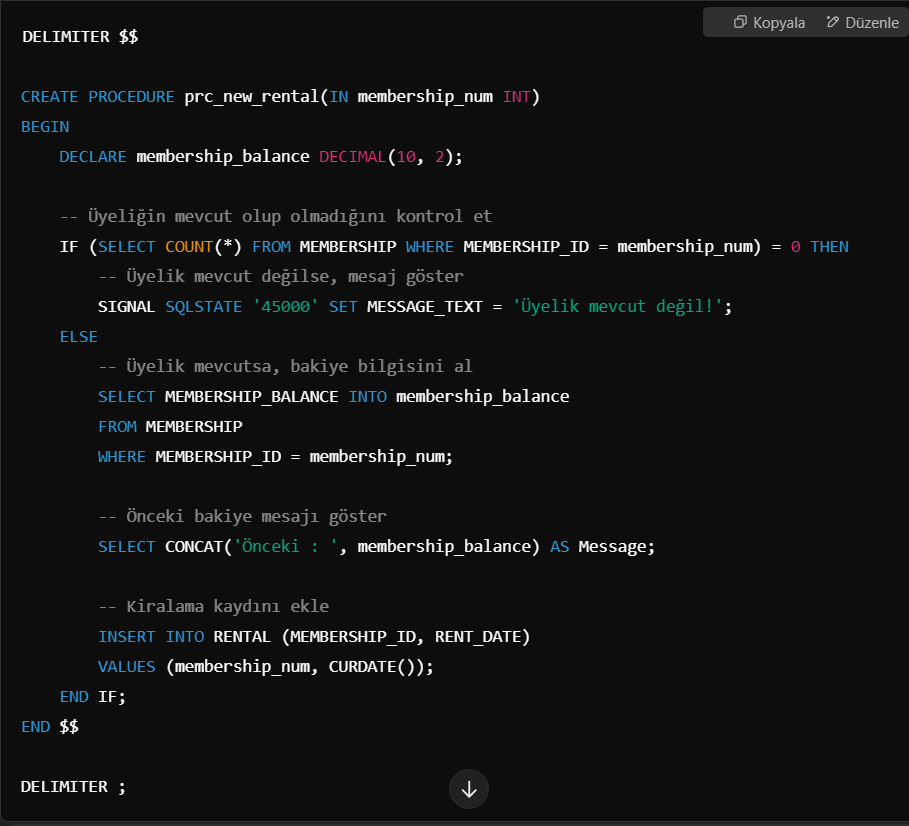
• Üyelik mevcutsa, üyelik alın ve bakiye tutarının önceki bakiye olduğunu belirten bir mesaj görüntüleyin. (Örneğin,

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.



· **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.