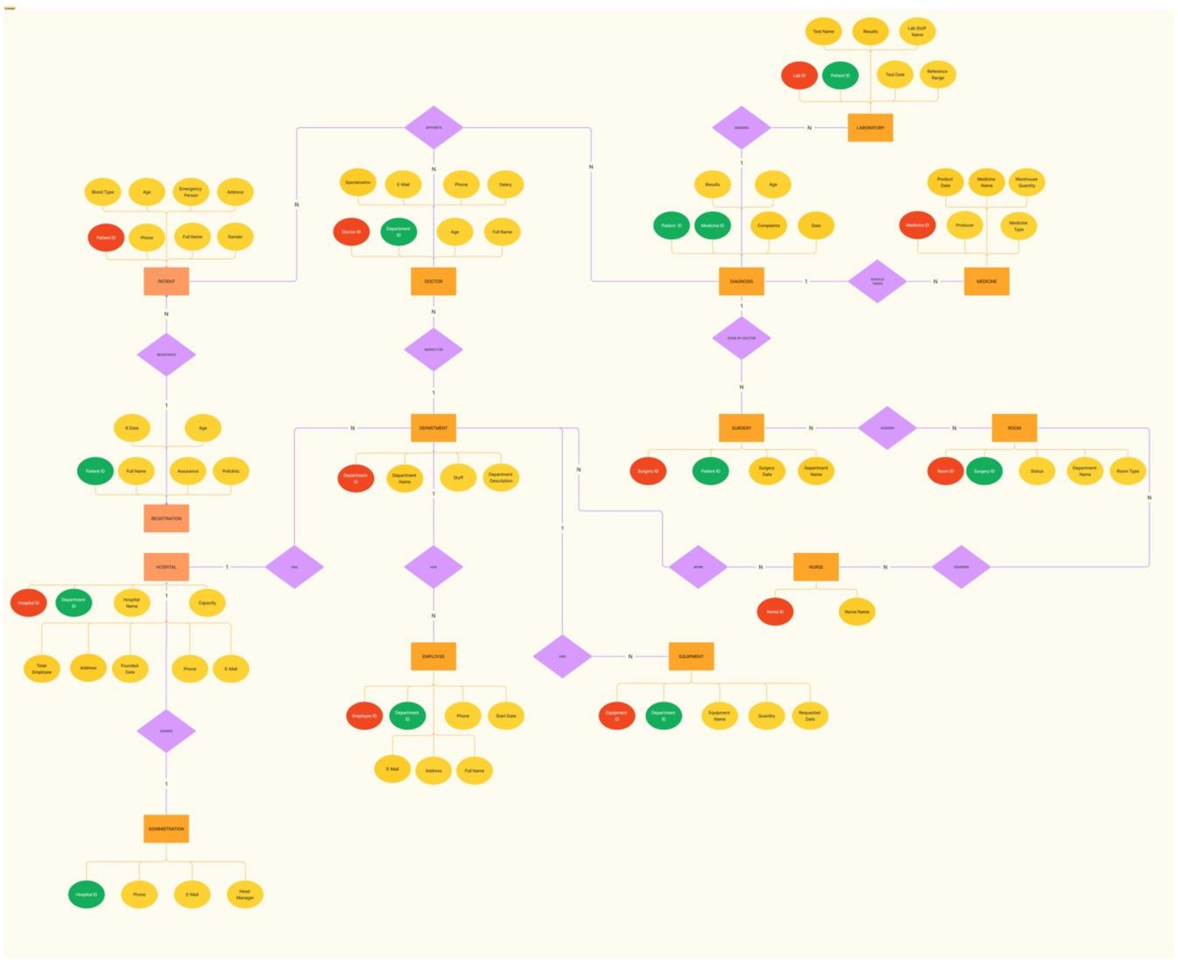
BİL312 VERİTABANI SİSTEMLERİ
Final Sınavı Projesi

Mehmet DAĞLI – 22120205382
Mert AKMAHMUT – 23120205082
Ahmet Nuri EROĞLU - 23120205087

Table of Contents

ERD GÖRÜNÜMÜ



Daha detaylı inceleme için aşağıdaki linki ziyaret edebilirsiniz:

https://www.figma.com/board/1gVUcxtfnftdlh0SiNkCAt/Untitled?node-id=2%3A13&t=W7PIGOkV5nOCdoBw-1

TABLOLARLA ILGILI GENEL BILGILER

Tablolar:

Registration (Kayıt) Tablosu: Hasta kayıt bilgilerini tutar.
Patient_ID: Yabancı anahtar (Foreign Key), Patient tablosuna referans verir.
Full_Name: Tam adı
Assurance: Güvence
Policlinic: Poliklinik
Age: Yaş
R_Date: Kayıt tarihi

Patient (Hasta) Tablosu: Hasta bilgilerini tutar.
Patient_ID: Birincil anahtar (Primary Key)
Phone: Telefon numarası
Full_Name: Tam adı
Gender: Cinsiyet
Blood_Type: Kan grubu
Address: Adres
Emergency_Person: Acil durumda aranacak kişi
Age: Yaş

Doctor (Doktor) Tablosu: Doktor bilgilerini tutar.
Doctor_ID: Birincil anahtar (Primary Key)
Department_ID: Yabancı anahtar (Foreign Key), Department tablosuna referans verir.
Age: Yaş
Full_Name: Tam adı
Salary: Maaş
Phone: Telefon numarası
Email: E-posta
Specialization: Uzmanlık alanı

Diagnosis (Teşhis) Tablosu: Teşhis bilgilerini tutar.
Patient_ID: Yabancı anahtar (Foreign Key)
Medicine_ID: Yabancı anahtar (Foreign Key), Medicine tablosuna referans verir.
Complaints: Şikayetler
Diagnosis_Date: Teşhis tarihi
Age: Yaş
Results: Sonuçlar

Appoints hereket tablosu: Doctor ,Patient ve Diagnosis varlık tablolarını birbirine bağlar. bu tabloda patient ID ve doctor ID foreign keydir. ayrıca bu tabloda Appointment Date ve Appointment Time diğer öznitelikler olarak tasarlanmıştır.

Medicine (İlaç) Tablosu: İlaç bilgilerini tutar.
Medicine_ID: Birincil anahtar (Primary Key)

Producer: Üretici
Medicine_Type: İlaç türü
Warehouse_Quantity: Depo miktarı
Medicine_Name: İlaç adı
Product_Date: Üretim tarihi

Laboratory (Laboratuvar) Tablosu: Laboratuvar test bilgilerini tutar.
Lab_ID: Birincil anahtar (Primary Key)
Patient_ID: Yabancı anahtar (Foreign Key), Doctor tablosuna referans verir.
Test_Date: Test tarihi
Reference_Range: Referans aralığı
Lab_Stuff_Name: Laboratuvar personeli adı
Results: Sonuçlar
Test_Name: Test adı

Surgery (Ameliyat) Tablosu: Ameliyat bilgilerini tutar.
Surgery_ID: Birincil anahtar (Primary Key)
Patient_ID: Yabancı anahtar (Foreign Key), Doctor tablosuna referans verir.
Surgery_Date: Ameliyat tarihi

Room (Oda) Tablosu: Oda bilgilerini tutar.
Room_ID: Birincil anahtar (Primary Key)
Surgery_ID: Yabancı anahtar (Foreign Key), Surgery tablosuna referans verir.
Status: Durum
Department_Name: Departman adı
Room_Type: Oda türü

Nurse (Hemşire) Tablosu: Hemşire bilgilerini tutar.
Nurse_ID: Birincil anahtar (Primary Key)
Nurse_Name: Hemşire adı

Work hareket tablosu: Nurse ile Department arasında ilişki kurar. bu tabloda Nurse ID ve Department ID Foreign key olarak tutulmuştur.

Governs hareket tablosu : Nurse ile Room arasında ilişki kurar. bu tabloda Nurse ID ve Room ID Foreign Key olarak tutulmaktadır.

Hospital (Hastane) Tablosu: Hastane bilgilerini tutar.
Hospital_ID: Birincil anahtar (Primary Key)
Hospital_Name: Hastane adı
Capacity: Kapasite
Phone: Telefon numarası
Total_Employee: Toplam çalışan sayısı
Address: Adres
Founded_Date: Kuruluş tarihi
Email: E-posta

Departent_ID: Department tablosuna referans verir(Foreign Key)

Administration (Yönetim) Tablosu: Yönetim bilgilerini tutar.
Hospital_ID: Yabancı anahtar (Foreign Key), Hospital tablosuna referans verir.
Phone: Telefon numarası
Head_Manager: Baş yönetici
Email: E-posta

Department (Departman) Tablosu: Departman bilgilerini tutar.
Department_ID: Birincil anahtar (Primary Key)
Department_Name: Departman adı
Stuff: Malzemeler
Department_Description: Departman açıklaması

Employee (Çalışan) Tablosu: Çalışan bilgilerini tutar.
Employee_ID: Birincil anahtar (Primary Key)
Department_ID: Yabancı anahtar (Foreign Key), Department tablosuna referans verir.
Phone: Telefon numarası
Start_Date: Başlangıç tarihi
Email: E-posta
Address: Adres
Full_Name: Tam adı

Equipment (Ekipman) Tablosu: Ekipman bilgilerini tutar.
Equipment_ID: Birincil anahtar (Primary Key)
Department_ID: Yabancı anahtar (Foreign Key), Department tablosuna referans verir.
Equipment_Name: Ekipman adı
Equipment_Quantity: Ekipman miktarı
Requested_Date: Talep tarihi

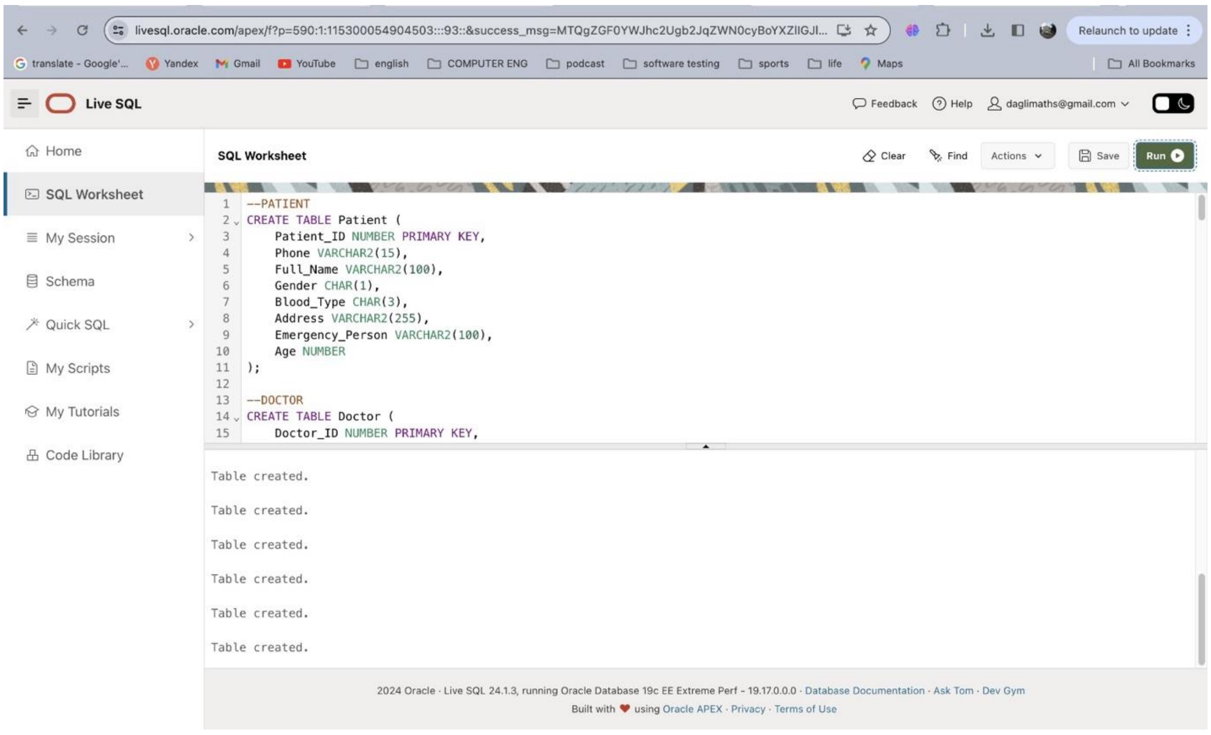
TABLO OLUŞTURMA KODLARI

1)PROJENİN TABLO OLUŞTURMA KODLARININ YAZILMASI

```
--PATIENT
CREATE TABLE Patient (
   Patient_ID NUMBER PRIMARY KEY,
   Phone VARCHAR2(15),
   Full_Name VARCHAR2(100),
   Gender VARCHAR2(10),
   Blood_Type VARCHAR2(5),
   Address VARCHAR2(200),
   Emergency_Person VARCHAR2(100),
   Age NUMBER
);
--DOCTOR
CREATE TABLE Doctor (
   Doctor_ID NUMBER PRIMARY KEY,
   Department_ID NUMBER,
   Age NUMBER,
   Full_Name VARCHAR2(100),
   Salary NUMBER,
   Phone VARCHAR2(15),
   Email VARCHAR2(100),
   Specialization VARCHAR2(100),
   CONSTRAINT fk_doctor_patient FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID),
   CONSTRAINT fk_doctor_department FOREIGN KEY (Department_ID) REFERENCES
Department(Department_ID)
);
--REGISTRATION
CREATE TABLE Registration (
   Patient_ID NUMBER,
   Full_Name VARCHAR2(100),
   Assurance VARCHAR2(100),
   Policlinic VARCHAR2(100),
   Age NUMBER,
   R_Date DATE,
   CONSTRAINT fk_registration_patient FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID)
);
--HOSPİTAL
CREATE TABLE Hospital (
   Hospital_ID NUMBER PRIMARY KEY,
   Hospital_Name VARCHAR2(100),
   Capacity NUMBER,
   Phone VARCHAR2(15),
   Total_Employee NUMBER,
   Address VARCHAR2(200),
   Founded_Date DATE,
   Email VARCHAR2(100),
   Department_ID NUMBER,
   CONSTRAINT fk_hospital_department FOREIGN KEY (Department_ID) REFERENCES
Department(Department_ID)
);
--ADMINISTRATION
CREATE TABLE Administration (
   Hospital_ID NUMBER,
   Phone VARCHAR2(15),
   Head_Manager VARCHAR2(100),
   Email VARCHAR2(100),
   CONSTRAINT fk_administration_hospital FOREIGN KEY (Hospital_ID) REFERENCES
Hospital(Hospital_ID)
);
--DIAGNOSIS
CREATE TABLE Diagnosis (
   Patient_ID NUMBER,
   Medicine_ID NUMBER,
   Complaints VARCHAR2(500),
   Diagnosis_Date DATE,
   Age NUMBER,
   Results VARCHAR2(500),
   CONSTRAINT fk_diagnosis_patient FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID),
   CONSTRAINT fk_diagnosis_medicine FOREIGN KEY (Medicine_ID) REFERENCES
Medicine(Medicine_ID)
);
--APPOINTS MOVEMENT TABLE
CREATE TABLE Appoints (
   Patient_ID NUMBER,
   Doctor_ID NUMBER,
   Appointment_Date DATE,
   Appointment_Time DATE,
   CONSTRAINT fk_appoints_patient FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID),
   CONSTRAINT fk_appoints_doctor FOREIGN KEY (Doctor_ID) REFERENCES
Doctor(Doctor_ID)
);
--MEDICINE
CREATE TABLE Medicine (
   Medicine_ID NUMBER PRIMARY KEY,
   Producer VARCHAR2(100),
   Medicine_Type VARCHAR2(100),
   Warehouse_Quantity NUMBER,
   Medicine_Name VARCHAR2(100),
   Product_Date DATE
);
--LABORATORY
CREATE TABLE Laboratory (
   Lab_ID NUMBER PRIMARY KEY,
   Patient_ID NUMBER,
   Test_Date DATE,
   Reference_Range VARCHAR2(100),
   Lab_Stuff_Name VARCHAR2(100),
   Results VARCHAR2(500),
   Test_Name VARCHAR2(100),
   CONSTRAINT fk_laboratory_patient FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID)
);
--SURGERY
CREATE TABLE Surgery (
   Surgery_ID NUMBER PRIMARY KEY,
   Patient_ID NUMBER,
   Surgery_Date DATE,
   CONSTRAINT fk_surgery_patient FOREIGN KEY (Patient_ID) REFERENCES
Patient(Patient_ID)
);
--ROOM
CREATE TABLE Room (
   Room_ID NUMBER PRIMARY KEY,
   Surgery_ID NUMBER,
   Status VARCHAR2(100),
   Department_Name VARCHAR2(100),
   Room_Type VARCHAR2(100),
   CONSTRAINT fk_room_surgery FOREIGN KEY (Surgery_ID) REFERENCES
Surgery(Surgery_ID)
);
--NURSE
CREATE TABLE Nurse (
   Nurse_ID NUMBER PRIMARY KEY,
   Nurse_Name VARCHAR2(100)
);
--GOVERNS MOVEMENT TABLE
CREATE TABLE Governs (
   Nurse_ID NUMBER,
   Room_ID NUMBER,
   CONSTRAINT fk_governs_nurse FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID),
   CONSTRAINT fk_governs_room FOREIGN KEY (Room_ID) REFERENCES Room(Room_ID)
);
```

```
--WORK MOVEMENT TABLE
CREATE TABLE Work (
   Nurse_ID NUMBER,
   Department_ID NUMBER,
   CONSTRAINT fk_work_nurse FOREIGN KEY (Nurse_ID) REFERENCES Nurse(Nurse_ID),
   CONSTRAINT fk_work_department FOREIGN KEY (Department_ID) REFERENCES
Department(Department_ID)
);
--DEPARTMENT
CREATE TABLE Department (
   Department_ID NUMBER PRIMARY KEY,
   Department_Name VARCHAR2(100),
   Stuff VARCHAR2(100),
   Department_Description VARCHAR2(500)
);
--EMPLOYEE
CREATE TABLE Employee (
   Employee_ID NUMBER PRIMARY KEY,
   Department_ID NUMBER,
   Phone VARCHAR2(15),
   Start_Date DATE,
   Email VARCHAR2(100),
   Address VARCHAR2(200),
   Full_Name VARCHAR2(100),
   CONSTRAINT fk_employee_department FOREIGN KEY (Department_ID) REFERENCES
Department(Department_ID)
);
--EQUIPMENT
CREATE TABLE Equipment (
   Equipment_ID NUMBER PRIMARY KEY,
   Department_ID NUMBER,
   Equipment_Name VARCHAR2(100),
   Equipment_Quantity NUMBER,
   Requested_Date DATE,
   CONSTRAINT fk_equipment_department FOREIGN KEY (Department_ID) REFERENCES
Department(Department_ID)
);
```

ÇIKTI:



```
--PATIENT
CREATE TABLE Patient (
    Patient_ID NUMBER PRIMARY KEY,
    Phone VARCHAR2(15),
    Full_Name VARCHAR2(100),
    Gender CHAR(1),
    Blood_Type CHAR(3),
    Address VARCHAR2(255),
    Emergency_Person VARCHAR2(100),
    Age NUMBER
);

--DOCTOR
CREATE TABLE Doctor (
    Doctor_ID NUMBER PRIMARY KEY,
```

```
Table created.

Table created.

Table created.

Table created.

Table created.

Table created.
```

2024 Oracle · Live SQL 24.1.3, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with ♥ using Oracle APEX · Privacy · Terms of Use

2) PROJENIZIN INDEKSLERINI(BIRINCIL, YABANCI VEYA BILEŞIK ANAHTARLAR) OLUŞTURAN KODLARI GELIŞTIRINIZ.

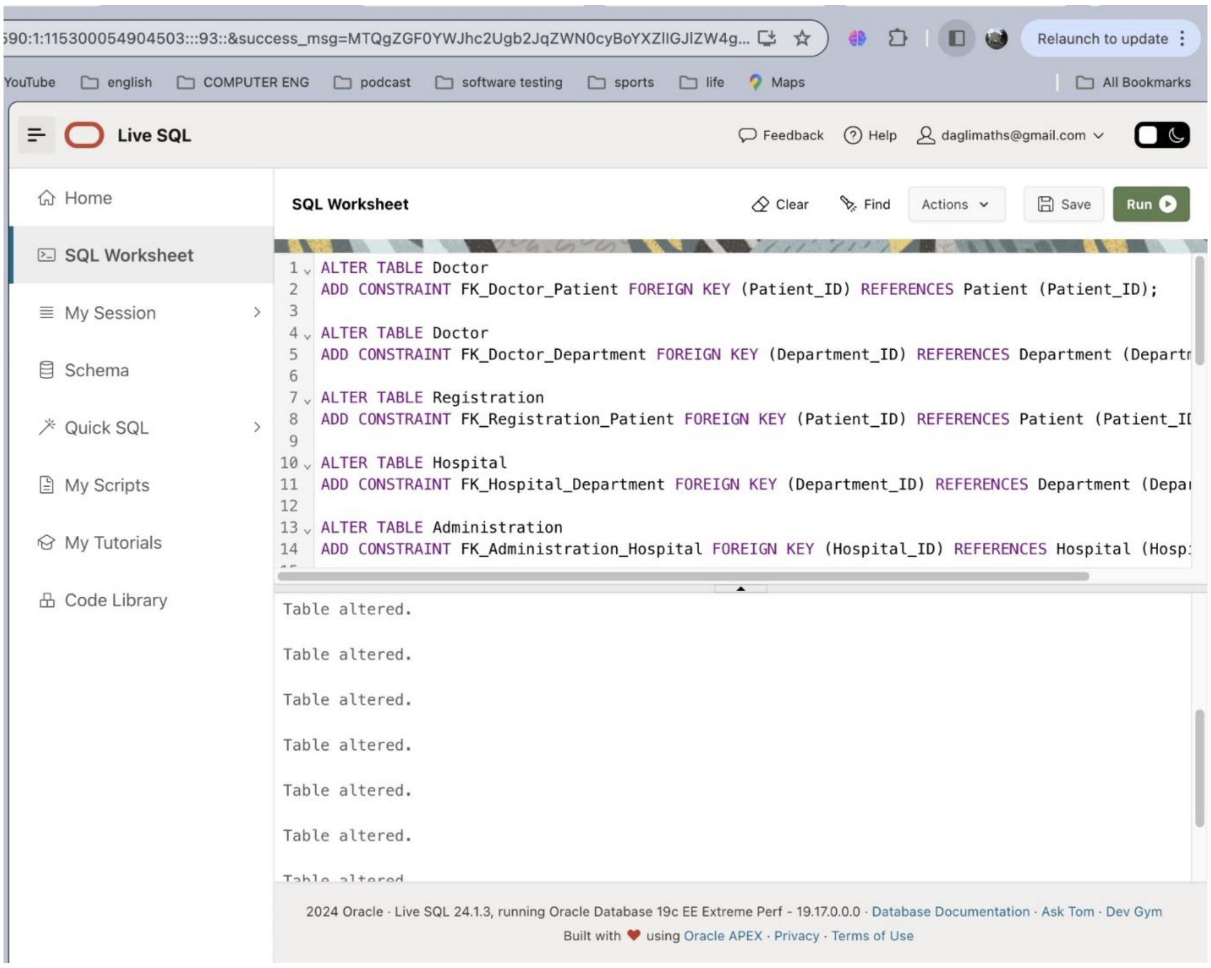
```
-- Varlık Tabloları için Foreign Key Oluşturma
-- Patient tablosu için
ALTER TABLE Registration
ADD CONSTRAINT fk_registration_patient
FOREIGN KEY (Patient_ID)
REFERENCES Patient(Patient_ID);

-- Doctor tablosu için
ALTER TABLE Diagnosis
ADD CONSTRAINT fk_diagnosis_doctor
FOREIGN KEY (Patient_ID)
REFERENCES Patient(Patient_ID);

ALTER TABLE Surgery
ADD CONSTRAINT fk_surgery_doctor
FOREIGN KEY (Patient_ID)
REFERENCES Patient(Patient_ID);

-- Department tablosu için
```

```
ALTER TABLE Hospital
ADD CONSTRAINT fk_hospital_department
FOREIGN KEY (Department_ID)
REFERENCES Department(Department_ID);

-- Medicine tablosu için (Hareket tablosu)
ALTER TABLE Diagnosis
ADD CONSTRAINT fk_diagnosis_medicine
FOREIGN KEY (Medicine_ID)
REFERENCES Medicine(Medicine_ID);

-- Nurse tablosu için (Hareket tablosu)
ALTER TABLE Governs
ADD CONSTRAINT fk_governs_nurse
FOREIGN KEY (Nurse_ID)
REFERENCES Nurse(Nurse_ID);

-- Equipment tablosu için (Hareket tablosu)
ALTER TABLE Equipment
ADD CONSTRAINT fk_equipment_department
FOREIGN KEY (Department_ID)
REFERENCES Department(Department_ID);

-- Hareket Tabloları için Foreign Key Oluşturma
-- Appoints tablosu için
ALTER TABLE Appoints
ADD CONSTRAINT fk_appoints_patient
FOREIGN KEY (Patient_ID)
REFERENCES Patient(Patient_ID);

ALTER TABLE Appoints
ADD CONSTRAINT fk_appoints_doctor
FOREIGN KEY (Doctor_ID)
REFERENCES Doctor(Doctor_ID);

-- Work tablosu için
ALTER TABLE Work
ADD CONSTRAINT fk_work_nurse
FOREIGN KEY (Nurse_ID)
REFERENCES Nurse(Nurse_ID);

ALTER TABLE Work
ADD CONSTRAINT fk_work_department
FOREIGN KEY (Department_ID)
REFERENCES Department(Department_ID);

-- Governs tablosu için
```

ALTER TABLE Governs
ADD CONSTRAINT fk_governs_nurse
FOREIGN KEY (Nurse_ID)
REFERENCES Nurse(Nurse_ID);

ALTER TABLE Governs
ADD CONSTRAINT fk_governs_room
FOREIGN KEY (Room_ID)
REFERENCES Room(Room_ID);

ÇIKTI:



```
ALTER TABLE Doctor
ADD CONSTRAINT FK_Doctor_Patient FOREIGN KEY (Patient_ID) REFERENCES Patient (Patient_ID);

ALTER TABLE Doctor
ADD CONSTRAINT FK_Doctor_Department FOREIGN KEY (Department_ID) REFERENCES Department (Departm

ALTER TABLE Registration
ADD CONSTRAINT FK_Registration_Patient FOREIGN KEY (Patient_ID) REFERENCES Patient (Patient_ID

ALTER TABLE Hospital
ADD CONSTRAINT FK_Hospital_Department FOREIGN KEY (Department_ID) REFERENCES Department (Depar

ALTER TABLE Administration
ADD CONSTRAINT FK_Administration_Hospital FOREIGN KEY (Hospital_ID) REFERENCES Hospital (Hosp
```

```
Table altered.

Table altered.

Table altered.

Table altered.

Table altered.

Table altered.

Table altered.
```

2024 Oracle · Live SQL 24.1.3, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with ❤ using Oracle APEX · Privacy · Terms of Use

3) PROJENIZIN VERI GIRIŞINI BIR PL/SQL PAKETTEN ÇAĞRILABILEN BIR KOD ARACILIĞI(PROSEDÜR VEYA FONKSIYON) ILE GERÇEKLEŞTIRINIZ.

```
CREATE OR REPLACE PACKAGE Veri_Girisi_Package IS
  -- Prosedür tanımlamaları
  PROCEDURE Kayit_Ekle (
    p_patient_id IN Registration.Patient_ID%TYPE,
    p_full_name IN Registration.Full_Name%TYPE,
    p_assurance IN Registration.Assurance%TYPE,
    p_policlinic IN Registration.Policlinic%TYPE,
    p_age IN Registration.Age%TYPE,
    p_r_date IN Registration.R_Date%TYPE
  );

  PROCEDURE Hasta_Ekle (
    p_patient_id IN Patient.Patient_ID%TYPE,
    p_phone IN Patient.Phone%TYPE,
    p_full_name IN Patient.Full_Name%TYPE,
    p_gender IN Patient.Gender%TYPE,
    p_blood_type IN Patient.Blood_Type%TYPE,
    p_address IN Patient.Address%TYPE,
    p_emergency_person IN Patient.Emergency_Person%TYPE,
    p_age IN Patient.Age%TYPE
  );

  PROCEDURE Doktor_Ekle (
    p_doctor_id IN Doctor.Doctor_ID%TYPE,
    p_department_id IN Doctor.Department_ID%TYPE,
    p_age IN Doctor.Age%TYPE,
    p_full_name IN Doctor.Full_Name%TYPE,
    p_salary IN Doctor.Salary%TYPE,
    p_phone IN Doctor.Phone%TYPE,
    p_email IN Doctor.Email%TYPE,
    p_specialization IN Doctor.Specialization%TYPE
  );

  PROCEDURE Teshis_Ekle (
    p_patient_id IN Diagnosis.Patient_ID%TYPE,
    p_medicine_id IN Diagnosis.Medicine_ID%TYPE,
    p_complaints IN Diagnosis.Complaints%TYPE,
    p_diagnosis_date IN Diagnosis.Diagnosis_Date%TYPE,
    p_age IN Diagnosis.Age%TYPE,
    p_results IN Diagnosis.Results%TYPE
  );
```

```
PROCEDURE Randevu_Ekle (
   p_patient_id IN Appoints.Patient_ID%TYPE,
   p_doctor_id IN Appoints.Doctor_ID%TYPE,
   p_appointment_date IN Appoints.Appointment_Date%TYPE,
   p_appointment_time IN Appoints.Appointment_Time%TYPE
);

PROCEDURE Ilac_Ekle (
   p_medicine_id IN Medicine.Medicine_ID%TYPE,
   p_producer IN Medicine.Producer%TYPE,
   p_medicine_type IN Medicine.Medicine_Type%TYPE,
   p_warehouse_quantity IN Medicine.Warehouse_Quantity%TYPE,
   p_medicine_name IN Medicine.Medicine_Name%TYPE,
   p_product_date IN Medicine.Product_Date%TYPE
);

PROCEDURE Laboratuvar_Ekle (
   p_lab_id IN Laboratory.Lab_ID%TYPE,
   p_patient_id IN Laboratory.Patient_ID%TYPE,
   p_test_date IN Laboratory.Test_Date%TYPE,
   p_reference_range IN Laboratory.Reference_Range%TYPE,
   p_lab_stuff_name IN Laboratory.Lab_Stuff_Name%TYPE,
   p_results IN Laboratory.Results%TYPE,
   p_test_name IN Laboratory.Test_Name%TYPE
);

PROCEDURE Ameliyat_Ekle (
   p_surgery_id IN Surgery.Surgery_ID%TYPE,
   p_patient_id IN Surgery.Patient_ID%TYPE,
   p_surgery_date IN Surgery.Surgery_Date%TYPE
);

PROCEDURE Oda_Ekle (
   p_room_id IN Room.Room_ID%TYPE,
   p_surgery_id IN Room.Surgery_ID%TYPE,
   p_status IN Room.Status%TYPE,
   p_department_name IN Room.Department_Name%TYPE,
   p_room_type IN Room.Room_Type%TYPE
);

PROCEDURE Hemsire_Ekle (
   p_nurse_id IN Nurse.Nurse_ID%TYPE,
   p_nurse_name IN Nurse.Nurse_Name%TYPE
);

PROCEDURE Yonetir_Ekle (
    p_nurse_id IN Governs.Nurse_ID%TYPE,
    p_room_id IN Governs.Room_ID%TYPE
);

PROCEDURE Calisir_Ekle (
    p_nurse_id IN Work.Nurse_ID%TYPE,
    p_department_id IN Work.Department_ID%TYPE
);

PROCEDURE Departman_Ekle (
    p_department_id IN Department.Department_ID%TYPE,
    p_department_name IN Department.Department_Name%TYPE,
    p_stuff IN Department.Stuff%TYPE,
    p_department_description IN Department.Department_Description%TYPE
);

PROCEDURE Hastane_Ekle (
    p_hospital_id IN Hospital.Hospital_ID%TYPE,
    p_hospital_name IN Hospital.Hospital_Name%TYPE,
    p_capacity IN Hospital.Capacity%TYPE,
    p_phone IN Hospital.Phone%TYPE,
    p_total_employee IN Hospital.Total_Employee%TYPE,
    p_address IN Hospital.Address%TYPE,
    p_founded_date IN Hospital.Founded_Date%TYPE,
    p_email IN Hospital.Email%TYPE,
    p_department_id IN Hospital.Department_ID%TYPE
);

PROCEDURE Yonetim_Ekle (
    p_hospital_id IN Administration.Hospital_ID%TYPE,
    p_phone IN Administration.Phone%TYPE,
    p_head_manager IN Administration.Head_Manager%TYPE,
    p_email IN Administration.Email%TYPE
);

PROCEDURE Calisan_Ekle (
    p_employee_id IN Employee.Employee_ID%TYPE,
    p_department_id IN Employee.Department_ID%TYPE,
    p_phone IN Employee.Phone%TYPE,
    p_start_date IN Employee.Start_Date%TYPE,
    p_email IN Employee.Email%TYPE,
    p_address IN Employee.Address%TYPE,
    p_full_name IN Employee.Full_Name%TYPE
);

PROCEDURE Ekipman_Ekle (
    p_equipment_id IN Equipment.Equipment_ID%TYPE,
      p_department_id IN Equipment.Department_ID%TYPE,
      p_equipment_name IN Equipment.Equipment_Name%TYPE,
      p_equipment_quantity IN Equipment.Equipment_Quantity%TYPE,
      p_requested_date IN Equipment.Requested_Date%TYPE
   );
END Veri_Girisi_Package;
/
--body kısmı
CREATE OR REPLACE PACKAGE BODY Veri_Girisi_Package IS
   PROCEDURE Kayit_Ekle (
      p_patient_id IN Registration.Patient_ID%TYPE,
      p_full_name IN Registration.Full_Name%TYPE,
      p_assurance IN Registration.Assurance%TYPE,
      p_policlinic IN Registration.Policlinic%TYPE,
      p_age IN Registration.Age%TYPE,
      p_r_date IN Registration.R_Date%TYPE
   ) IS
   BEGIN
      INSERT INTO Registration (Patient_ID, Full_Name, Assurance, Policlinic, Age, R_Date)
      VALUES (p_patient_id, p_full_name, p_assurance, p_policlinic, p_age, p_r_date);
   END Kayit_Ekle;

   PROCEDURE Hasta_Ekle (
      p_patient_id IN Patient.Patient_ID%TYPE,
      p_phone IN Patient.Phone%TYPE,
      p_full_name IN Patient.Full_Name%TYPE,
      p_gender IN Patient.Gender%TYPE,
      p_blood_type IN Patient.Blood_Type%TYPE,
      p_address IN Patient.Address%TYPE,
      p_emergency_person IN Patient.Emergency_Person%TYPE,
      p_age IN Patient.Age%TYPE
   ) IS
   BEGIN
      INSERT INTO Patient (Patient_ID, Phone, Full_Name, Gender, Blood_Type, Address,
Emergency_Person, Age)
      VALUES (p_patient_id, p_phone, p_full_name, p_gender, p_blood_type, p_address,
p_emergency_person, p_age);
   END Hasta_Ekle;

   PROCEDURE Doktor_Ekle (
      p_doctor_id IN Doctor.Doctor_ID%TYPE,
      p_department_id IN Doctor.Department_ID%TYPE,
      p_age IN Doctor.Age%TYPE,
      p_full_name IN Doctor.Full_Name%TYPE,
      p_salary IN Doctor.Salary%TYPE,
      p_phone IN Doctor.Phone%TYPE,
      p_email IN Doctor.Email%TYPE,
      p_specialization IN Doctor.Specialization%TYPE
   ) IS
   BEGIN
      INSERT INTO Doctor (Doctor_ID, Department_ID, Age, Full_Name, Salary, Phone, Email,
Specialization)
      VALUES (p_doctor_id, p_department_id, p_age, p_full_name, p_salary, p_phone,
p_email, p_specialization);
   END Doktor_Ekle;

   PROCEDURE Teshis_Ekle (
      p_patient_id IN Diagnosis.Patient_ID%TYPE,
      p_medicine_id IN Diagnosis.Medicine_ID%TYPE,
      p_complaints IN Diagnosis.Complaints%TYPE,
      p_diagnosis_date IN Diagnosis.Diagnosis_Date%TYPE,
      p_age IN Diagnosis.Age%TYPE,
      p_results IN Diagnosis.Results%TYPE
   ) IS
   BEGIN
      INSERT INTO Diagnosis (Patient_ID, Medicine_ID, Complaints, Diagnosis_Date, Age,
Results)
      VALUES (p_patient_id, p_medicine_id, p_complaints, p_diagnosis_date, p_age,
p_results);
   END Teshis_Ekle;

   PROCEDURE Randevu_Ekle (
      p_patient_id IN Appoints.Patient_ID%TYPE,
      p_doctor_id IN Appoints.Doctor_ID%TYPE,
      p_appointment_date IN Appoints.Appointment_Date%TYPE,
      p_appointment_time IN Appoints.Appointment_Time%TYPE
   ) IS
   BEGIN
      INSERT INTO Appoints (Patient_ID, Doctor_ID, Appointment_Date, Appointment_Time)
      VALUES (p_patient_id, p_doctor_id, p_appointment_date, p_appointment_time);
   END Randevu_Ekle;

   PROCEDURE Ilac_Ekle (
      p_medicine_id IN Medicine.Medicine_ID%TYPE,
      p_producer IN Medicine.Producer%TYPE,
      p_medicine_type IN Medicine.Medicine_Type%TYPE,
      p_warehouse_quantity IN Medicine.Warehouse_Quantity%TYPE,
      p_medicine_name IN Medicine.Medicine_Name%TYPE,
      p_product_date IN Medicine.Product_Date%TYPE
   ) IS
   BEGIN
      INSERT INTO Medicine (Medicine_ID, Producer, Medicine_Type, Warehouse_Quantity,
Medicine_Name, Product_Date)
```

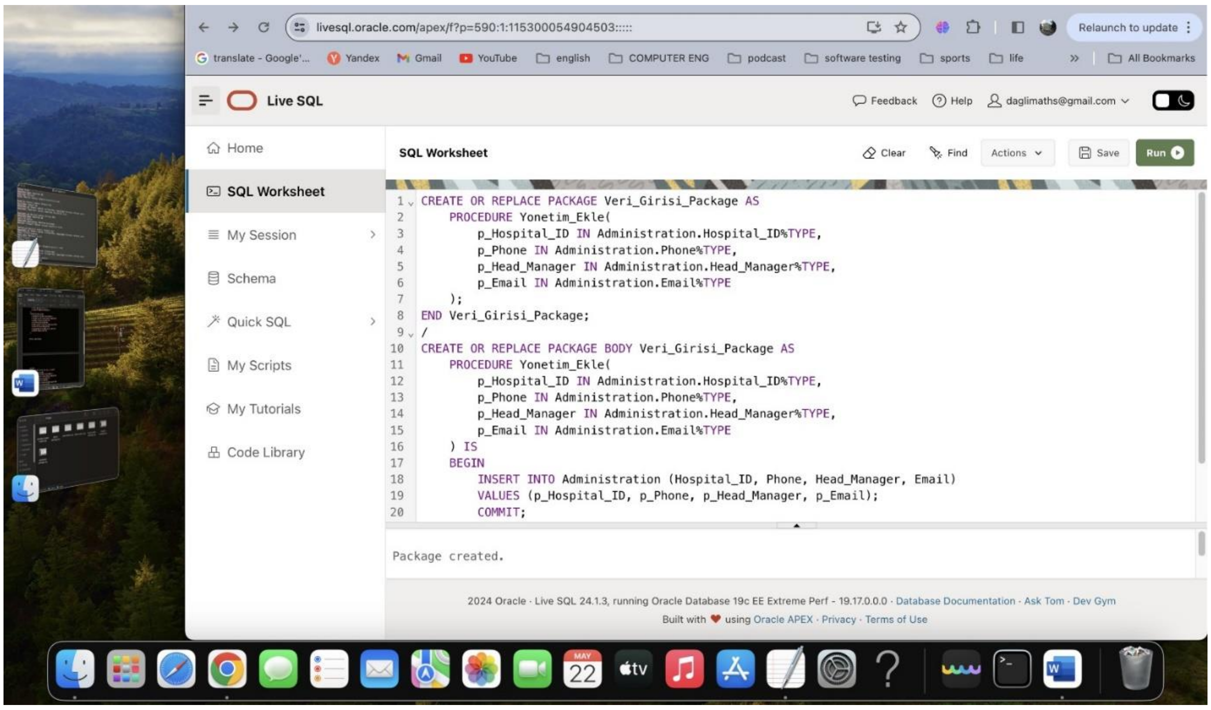
```
      VALUES (p_medicine_id, p_producer, p_medicine_type, p_warehouse_quantity,
p_medicine_name, p_product_date);
   END Ilac_Ekle;

   PROCEDURE Laboratuvar_Ekle (
      p_lab_id IN Laboratory.Lab_ID%TYPE,
      p_patient_id IN Laboratory.Patient_ID%TYPE,
      p_test_date IN Laboratory.Test_Date%TYPE,
      p_reference_range IN Laboratory.Reference_Range%TYPE,
      p_lab_stuff_name IN Laboratory.Lab_Stuff_Name%TYPE,
      p_results IN Laboratory.Results%TYPE,
      p_test_name IN Laboratory.Test_Name%TYPE
   ) IS
   BEGIN
      INSERT INTO Laboratory (Lab_ID, Patient_ID, Test_Date, Reference_Range,
Lab_Stuff_Name, Results, Test_Name)
      VALUES (p_lab_id, p_patient_id, p_test_date, p_reference_range, p_lab_stuff_name,
p_results, p_test_name);
   END Laboratuvar_Ekle;

   PROCEDURE Ameliyat_Ekle (
      p_surgery_id IN Surgery.Surgery_ID%TYPE,
      p_patient_id IN Surgery.Patient_ID%TYPE,
      p_surgery_date IN Surgery.Surgery_Date%TYPE
   ) IS
   BEGIN
      INSERT INTO Surgery (Surgery_ID, Patient_ID, Surgery_Date)
      VALUES (p_surgery_id, p_patient_id, p_surgery_date);
   END Ameliyat_Ekle;

   PROCEDURE Oda_Ekle (
      p_room_id IN Room.Room_ID%TYPE,
      p_surgery_id IN Room.Surgery_ID%TYPE,
      p_status IN Room.Status%TYPE,
      p_department_name IN Room.Department_Name%TYPE,
      p_room_type IN Room.Room_Type%TYPE
   ) IS
   BEGIN
      INSERT INTO Room (Room_ID, Surgery_ID, Status, Department_Name, Room_Type)
      VALUES (p_room_id, p_surgery_id, p_status, p_department_name, p_room_type);
   END Oda_Ekle;

   PROCEDURE Hemsire_Ekle (
      p_nurse_id IN Nurse.Nurse_ID%TYPE,
      p_nurse_name IN Nurse.Nurse_Name%TYPE
   ) IS
   BEGIN
```

```
        INSERT INTO Nurse (Nurse_ID, Nurse_Name)
        VALUES (p_nurse_id, p_nurse_name);
    END Hemsire_Ekle;

    PROCEDURE Yonetir_Ekle (
        p_nurse_id IN Governs.Nurse_ID%TYPE,
        p_room_id IN Governs.Room_ID%TYPE
    ) IS
    BEGIN
        INSERT INTO Governs (Nurse_ID, Room_ID)
        VALUES (p_nurse_id, p_room_id);
    END Yonetir_Ekle;

    PROCEDURE Calisir_Ekle (
        p_nurse_id IN Work.Nurse_ID%TYPE,
        p_department_id IN Work.Department_ID%TYPE
    ) IS
    BEGIN
        INSERT INTO Work (Nurse_ID, Department_ID)
        VALUES (p_nurse_id, p_department_id);
    END Calisir_Ekle;

    PROCEDURE Departman_Ekle (
        p_department_id IN Department.Department_ID%TYPE,
        p_department_name IN Department.Department_Name%TYPE,
        p_stuff IN Department.Stuff%TYPE,
        p_department_description IN Department.Department_Description%TYPE
    ) IS
    BEGIN
        INSERT INTO Department (Department_ID, Department_Name, Stuff,
Department_Description)
        VALUES (p_department_id, p_department_name, p_stuff, p_department_description);
    END Departman_Ekle;

    PROCEDURE Hastane_Ekle (
        p_hospital_id IN Hospital.Hospital_ID%TYPE,
        p_hospital_name IN Hospital.Hospital_Name%TYPE,
        p_capacity IN Hospital.Capacity%TYPE,
        p_phone IN Hospital.Phone%TYPE,
        p_total_employee IN Hospital.Total_Employee%TYPE,
        p_address IN Hospital.Address%TYPE,
        p_founded_date IN Hospital.Founded_Date%TYPE,
        p_email IN Hospital.Email%TYPE,
        p_department_id IN Hospital.Department_ID%TYPE
    ) IS
    BEGIN
```

```
      INSERT INTO Hospital (Hospital_ID, Hospital_Name, Capacity, Phone, Total_Employee,
Address, Founded_Date, Email, Department_ID)
      VALUES (p_hospital_id, p_hospital_name, p_capacity, p_phone, p_total_employee,
p_address, p_founded_date, p_email, p_department_id);
   END Hastane_Ekle;

   PROCEDURE Yonetim_Ekle (
      p_hospital_id IN Administration.Hospital_ID%TYPE,
      p_phone IN Administration.Phone%TYPE,
      p_head_manager IN Administration.Head_Manager%TYPE,
      p_email IN Administration.Email%TYPE
   ) IS
   BEGIN
      INSERT INTO Administration (Hospital_ID, Phone, Head_Manager, Email)
      VALUES (p_hospital_id, p_phone, p_head_manager, p_email);
   END Yonetim_Ekle;

   PROCEDURE Calisan_Ekle (
      p_employee_id IN Employee.Employee_ID%TYPE,
      p_department_id IN Employee.Department_ID%TYPE,
      p_phone IN Employee.Phone%TYPE,
      p_start_date IN Employee.Start_Date%TYPE,
      p_email IN Employee.Email%TYPE,
      p_address IN Employee.Address%TYPE,
      p_full_name IN Employee.Full_Name%TYPE
   ) IS
   BEGIN
      INSERT INTO Employee (Employee_ID, Department_ID, Phone, Start_Date, Email,
Address, Full_Name)
      VALUES (p_employee_id, p_department_id, p_phone, p_start_date, p_email,
p_address, p_full_name);
   END Calisan_Ekle;

   PROCEDURE Ekipman_Ekle (
      p_equipment_id IN Equipment.Equipment_ID%TYPE,
      p_department_id IN Equipment.Department_ID%TYPE,
      p_equipment_name IN Equipment.Equipment_Name%TYPE,
      p_equipment_quantity IN Equipment.Equipment_Quantity%TYPE,
      p_requested_date IN Equipment.Requested_Date%TYPE
   ) IS
   BEGIN
      INSERT INTO Equipment (Equipment_ID, Department_ID, Equipment_Name,
Equipment_Quantity, Requested_Date)
      VALUES (p_equipment_id, p_department_id, p_equipment_name,
p_equipment_quantity, p_requested_date);
   END Ekipman_Ekle;
END Veri_Girisi_Package;
```

```
/
-- örnek veri girişi
BEGIN
    Veri_Girisi_Package.Doktor_Ekle(
        p_doctor_id => 1,
        p_department_id => 2,
        p_age => 45,
        p_full_name => 'Dr. John Doe',
        p_salary => 150000,
        p_phone => '123-456-7890',
        p_email => 'johndoe@example.com',
        p_specialization => 'Cardiology'
    );
END;
```

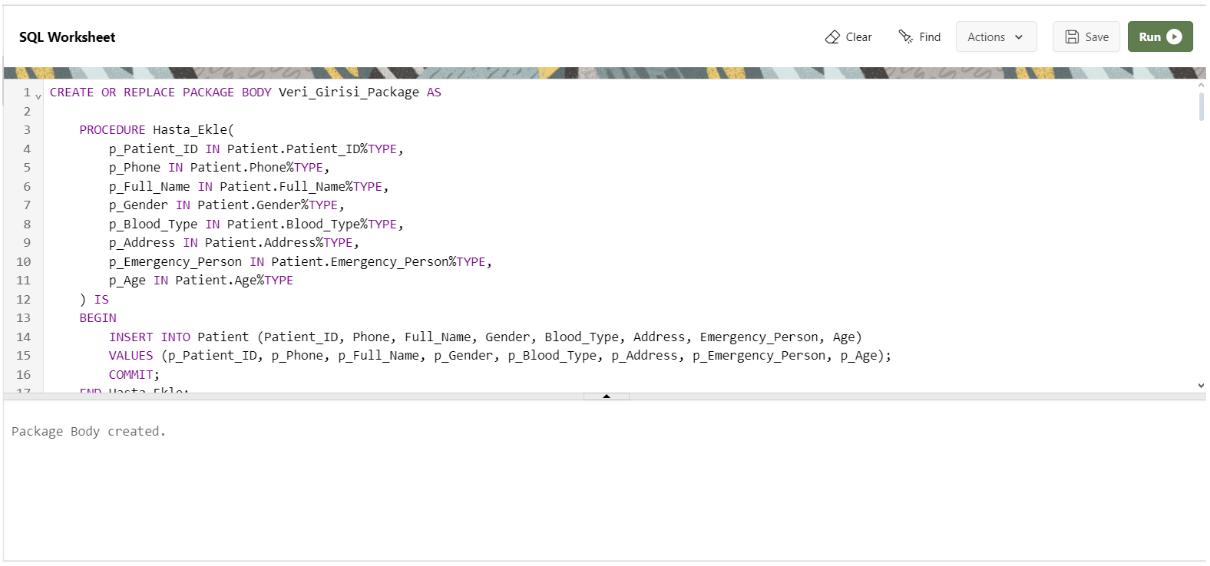
ÇIKTI:



```
CREATE OR REPLACE PACKAGE Veri_Girisi_Package AS
    PROCEDURE Yonetim_Ekle(
        p_Hospital_ID IN Administration.Hospital_ID%TYPE,
        p_Phone IN Administration.Phone%TYPE,
        p_Head_Manager IN Administration.Head_Manager%TYPE,
        p_Email IN Administration.Email%TYPE
    );
END Veri_Girisi_Package;
/
CREATE OR REPLACE PACKAGE BODY Veri_Girisi_Package AS
    PROCEDURE Yonetim_Ekle(
        p_Hospital_ID IN Administration.Hospital_ID%TYPE,
        p_Phone IN Administration.Phone%TYPE,
        p_Head_Manager IN Administration.Head_Manager%TYPE,
        p_Email IN Administration.Email%TYPE
    ) IS
    BEGIN
        INSERT INTO Administration (Hospital_ID, Phone, Head_Manager, Email)
        VALUES (p_Hospital_ID, p_Phone, p_Head_Manager, p_Email);
        COMMIT;
```

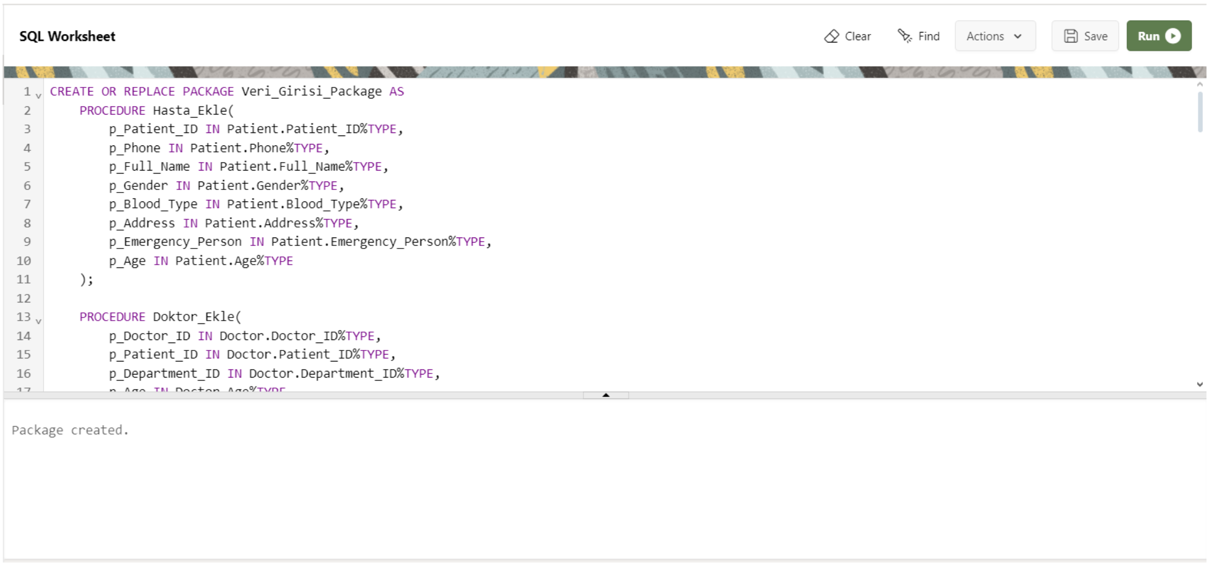
```
Package created.
```

2024 Oracle · Live SQL 24.1.3, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with ❤ using Oracle APEX · Privacy · Terms of Use



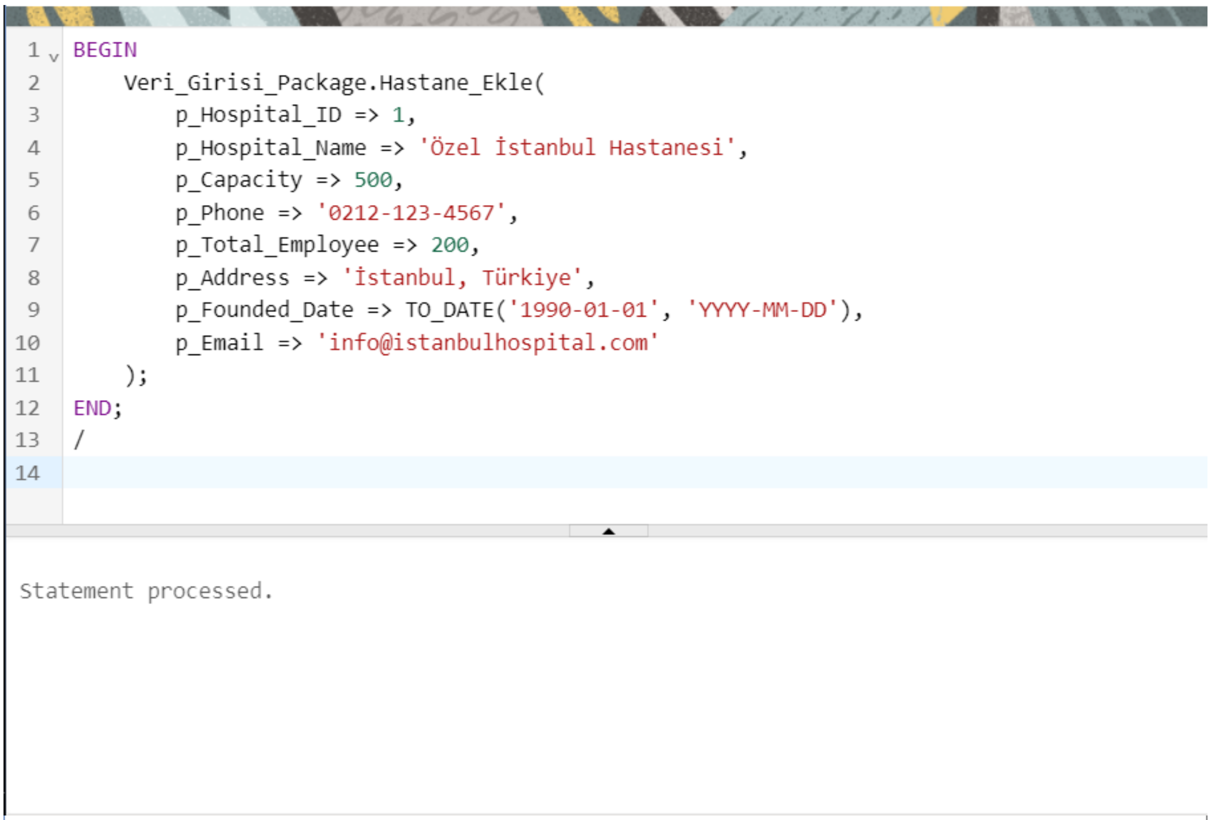
```
CREATE OR REPLACE PACKAGE BODY Veri_Girisi_Package AS

    PROCEDURE Hasta_Ekle(
        p_Patient_ID IN Patient.Patient_ID%TYPE,
        p_Phone IN Patient.Phone%TYPE,
        p_Full_Name IN Patient.Full_Name%TYPE,
        p_Gender IN Patient.Gender%TYPE,
        p_Blood_Type IN Patient.Blood_Type%TYPE,
        p_Address IN Patient.Address%TYPE,
        p_Emergency_Person IN Patient.Emergency_Person%TYPE,
        p_Age IN Patient.Age%TYPE
    ) IS
    BEGIN
        INSERT INTO Patient (Patient_ID, Phone, Full_Name, Gender, Blood_Type, Address, Emergency_Person, Age)
        VALUES (p_Patient_ID, p_Phone, p_Full_Name, p_Gender, p_Blood_Type, p_Address, p_Emergency_Person, p_Age);
        COMMIT;
    END Hasta_Ekle;
```

```
Package Body created.
```



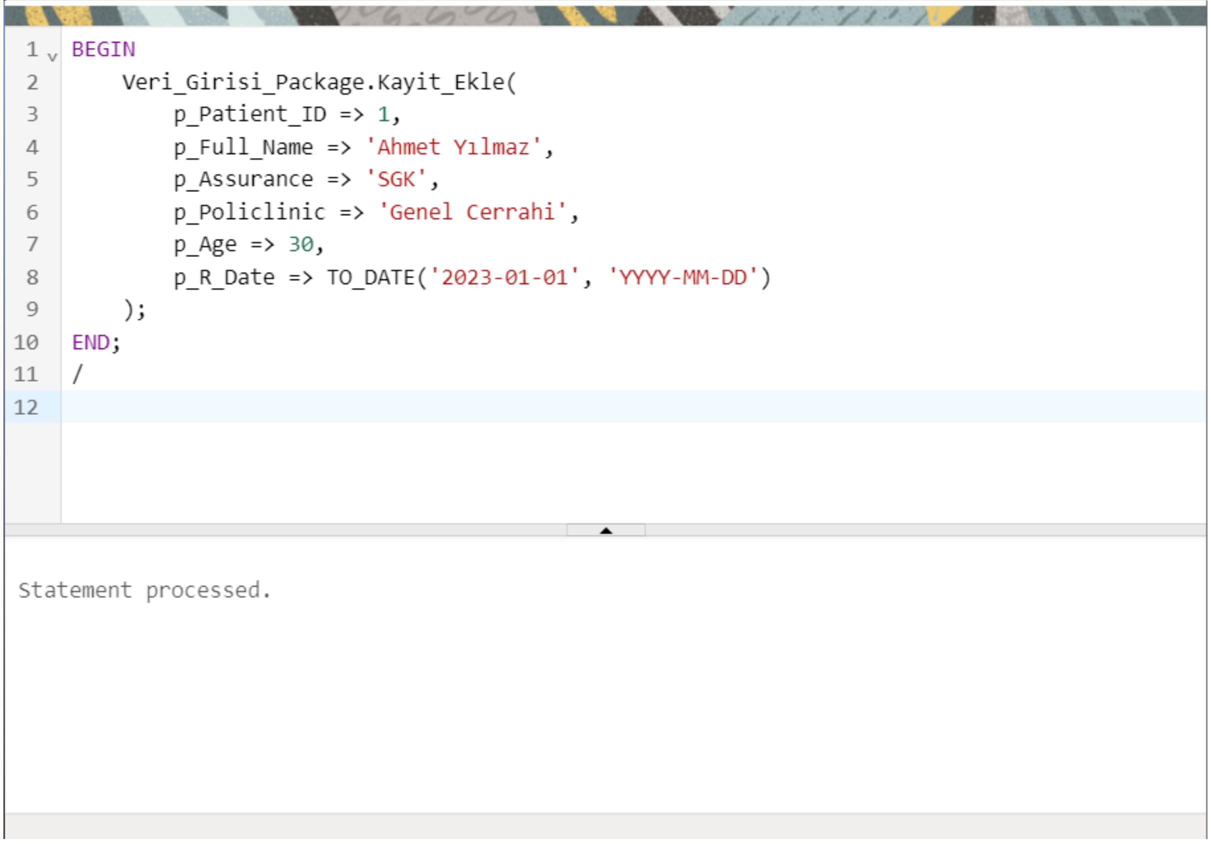
```
CREATE OR REPLACE PACKAGE Veri_Girisi_Package AS
    PROCEDURE Hasta_Ekle(
        p_Patient_ID IN Patient.Patient_ID%TYPE,
        p_Phone IN Patient.Phone%TYPE,
        p_Full_Name IN Patient.Full_Name%TYPE,
        p_Gender IN Patient.Gender%TYPE,
        p_Blood_Type IN Patient.Blood_Type%TYPE,
        p_Address IN Patient.Address%TYPE,
        p_Emergency_Person IN Patient.Emergency_Person%TYPE,
        p_Age IN Patient.Age%TYPE
    );

    PROCEDURE Doktor_Ekle(
        p_Doctor_ID IN Doctor.Doctor_ID%TYPE,
        p_Patient_ID IN Doctor.Patient_ID%TYPE,
        p_Department_ID IN Doctor.Department_ID%TYPE,
        p_Age IN Doctor.Age%TYPE
```

```
Package created.
```



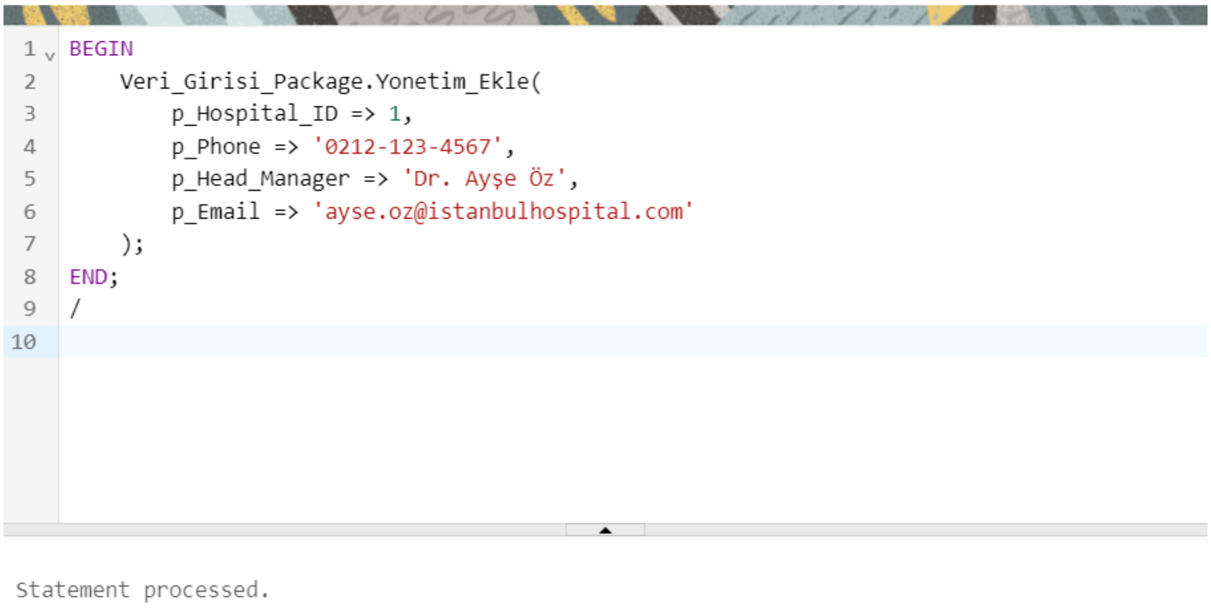
```
BEGIN
    Veri_Girisi_Package.Hastane_Ekle(
        p_Hospital_ID => 1,
        p_Hospital_Name => 'Özel İstanbul Hastanesi',
        p_Capacity => 500,
        p_Phone => '0212-123-4567',
        p_Total_Employee => 200,
        p_Address => 'İstanbul, Türkiye',
        p_Founded_Date => TO_DATE('1990-01-01', 'YYYY-MM-DD'),
        p_Email => 'info@istanbulhospital.com'
    );
END;
/
```

```
Statement processed.
```



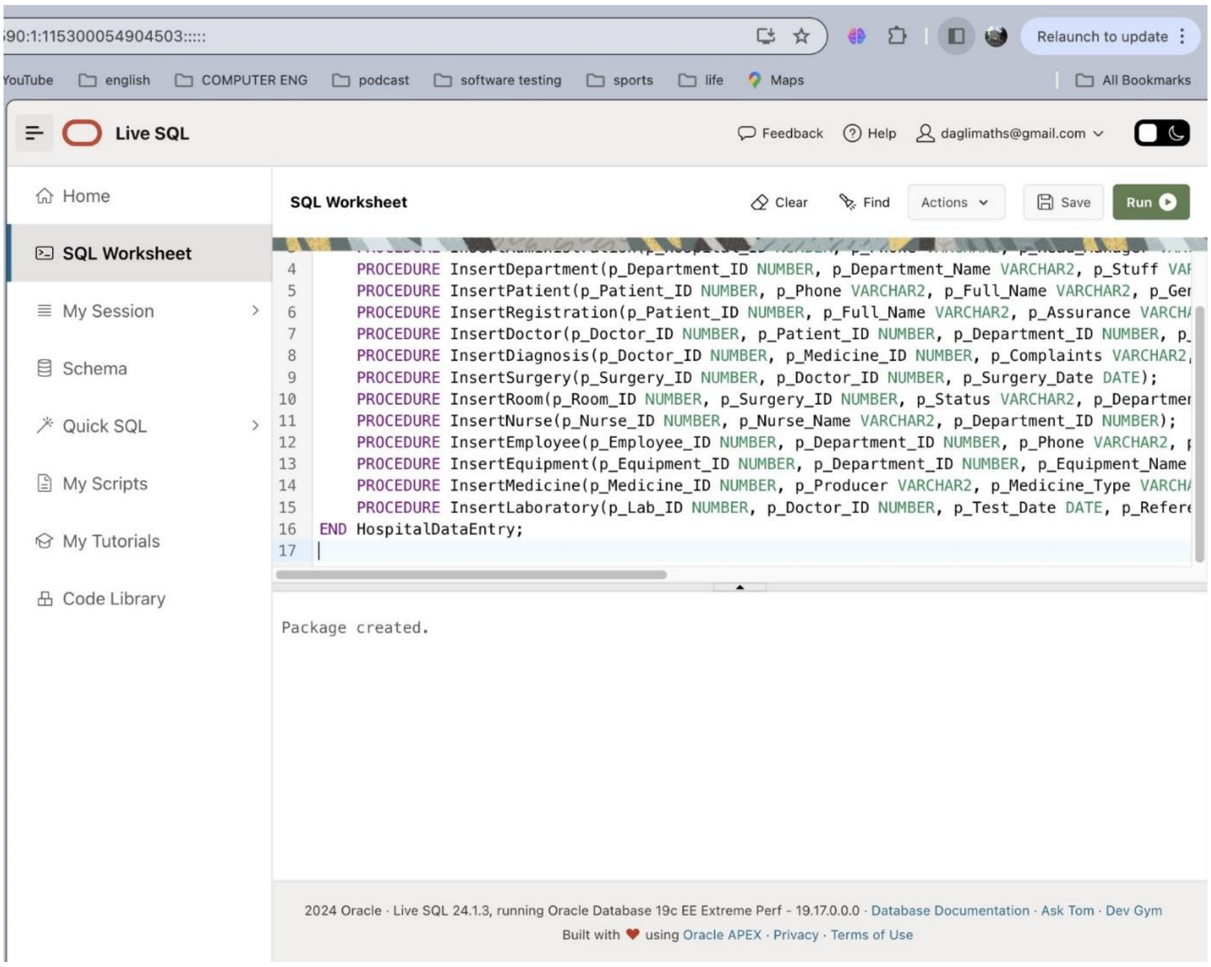
```
BEGIN
    Veri_Girisi_Package.Kayit_Ekle(
        p_Patient_ID => 1,
        p_Full_Name => 'Ahmet Yılmaz',
        p_Assurance => 'SGK',
        p_Policlinic => 'Genel Cerrahi',
        p_Age => 30,
        p_R_Date => TO_DATE('2023-01-01', 'YYYY-MM-DD')
    );
END;
/
```

```
Statement processed.
```



```
BEGIN
    Veri_Girisi_Package.Yonetim_Ekle(
        p_Hospital_ID => 1,
        p_Phone => '0212-123-4567',
        p_Head_Manager => 'Dr. Ayşe Öz',
        p_Email => 'ayse.oz@istanbulhospital.com'
    );
END;
/
```

```
Statement processed.
```



SQL Worksheet

```
    PROCEDURE InsertDepartment(p_Department_ID NUMBER, p_Department_Name VARCHAR2, p_Stuff VA
    PROCEDURE InsertPatient(p_Patient_ID NUMBER, p_Phone VARCHAR2, p_Full_Name VARCHAR2, p_Ge
    PROCEDURE InsertRegistration(p_Patient_ID NUMBER, p_Full_Name VARCHAR2, p_Assurance VARCH
    PROCEDURE InsertDoctor(p_Doctor_ID NUMBER, p_Patient_ID NUMBER, p_Department_ID NUMBER, p_
    PROCEDURE InsertDiagnosis(p_Doctor_ID NUMBER, p_Medicine_ID NUMBER, p_Complaints VARCHAR2,
    PROCEDURE InsertSurgery(p_Surgery_ID NUMBER, p_Doctor_ID NUMBER, p_Surgery_Date DATE);
    PROCEDURE InsertRoom(p_Room_ID NUMBER, p_Surgery_ID NUMBER, p_Status VARCHAR2, p_Departme
    PROCEDURE InsertNurse(p_Nurse_ID NUMBER, p_Nurse_Name VARCHAR2, p_Department_ID NUMBER);
    PROCEDURE InsertEmployee(p_Employee_ID NUMBER, p_Department_ID NUMBER, p_Phone VARCHAR2, 
    PROCEDURE InsertEquipment(p_Equipment_ID NUMBER, p_Department_ID NUMBER, p_Equipment_Name
    PROCEDURE InsertMedicine(p_Medicine_ID NUMBER, p_Producer VARCHAR2, p_Medicine_Type VARCH
    PROCEDURE InsertLaboratory(p_Lab_ID NUMBER, p_Doctor_ID NUMBER, p_Test_Date DATE, p_Refere
END HospitalDataEntry;
```

```
Package created.
```

2024 Oracle · Live SQL 24.1.3, running Oracle Database 19c EE Extreme Perf - 19.17.0.0.0 · Database Documentation · Ask Tom · Dev Gym
Built with ❤ using Oracle APEX · Privacy · Terms of Use

4)PROJENIZIN VERI GÜNCELLEMESINI BIR PL/SQL PAKETTEN ÇAĞRILABILEN BIR KOD ARACILIĞI(PROSEDÜR VEYA FONKSIYON) ILE GERÇEKLEŞTIRINIZ.

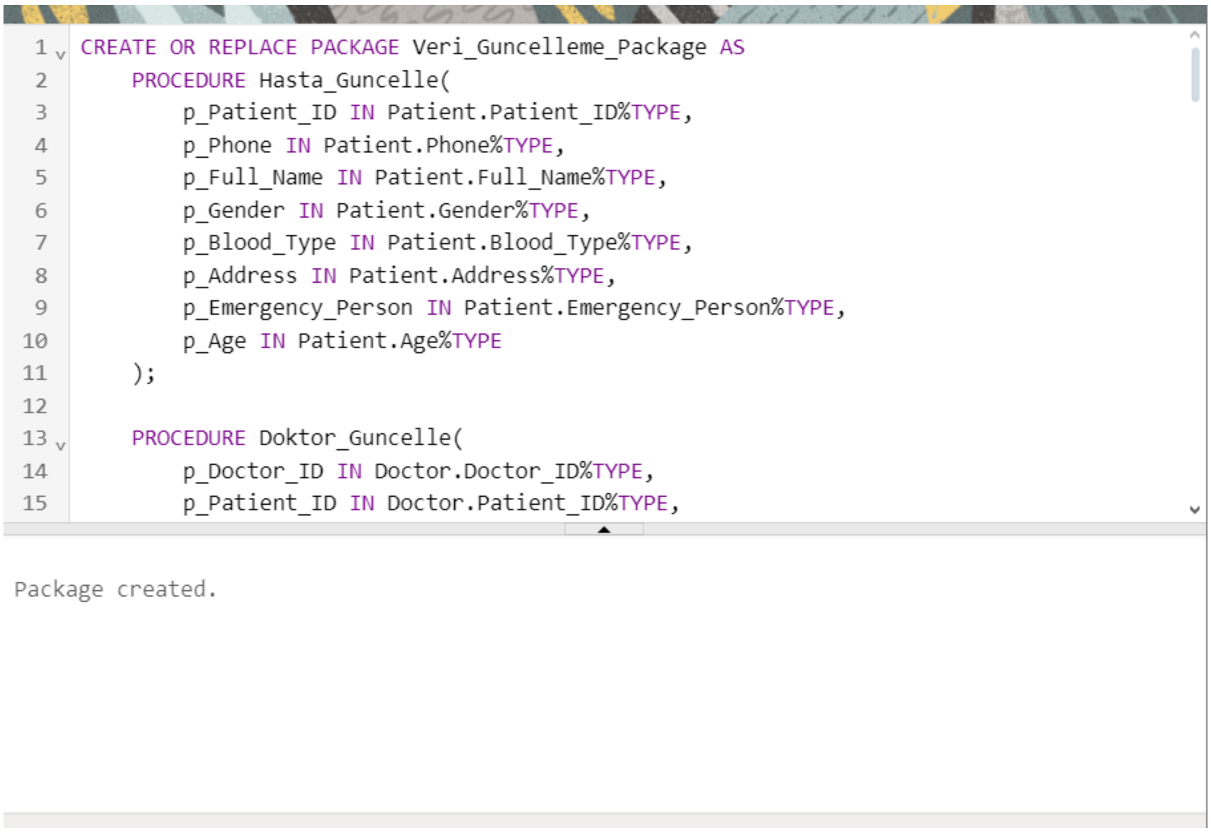
```
CREATE OR REPLACE PACKAGE Veri_Guncelleme_Package AS
   PROCEDURE Hasta_Guncelle (
      p_patient_id IN Patient.Patient_ID%TYPE,
      p_phone IN Patient.Phone%TYPE,
      p_full_name IN Patient.Full_Name%TYPE,
      p_gender IN Patient.Gender%TYPE,
      p_blood_type IN Patient.Blood_Type%TYPE,
      p_address IN Patient.Address%TYPE,
      p_emergency_person IN Patient.Emergency_Person%TYPE,
      p_age IN Patient.Age%TYPE
   );
  PROCEDURE Kayit_Guncelle(
      p_Patient_ID IN Registration.Patient_ID%TYPE,
      p_Full_Name IN Registration.Full_Name%TYPE,
      p_Assurance IN Registration.Assurance%TYPE,
      p_Policlinic IN Registration.Policlinic%TYPE,
      p_Age IN Registration.Age%TYPE,
      p_R_Date IN Registration.R_Date%TYPE
   );


  PROCEDURE Hastane_Guncelle(
      p_Hospital_ID IN Hospital.Hospital_ID%TYPE,
      p_Hospital_Name IN Hospital.Hospital_Name%TYPE,
      p_Capacity IN Hospital.Capacity%TYPE,
      p_Phone IN Hospital.Phone%TYPE,
      p_Total_Employee IN Hospital.Total_Employee%TYPE,
      p_Address IN Hospital.Address%TYPE,
      p_Founded_Date IN Hospital.Founded_Date%TYPE,
      p_Email IN Hospital.Email%TYPE
   );
   PROCEDURE Yonetim_Guncelle(
      p_Hospital_ID IN Administration.Hospital_ID%TYPE,
      p_Phone IN Administration.Phone%TYPE,
      p_Head_Manager IN Administration.Head_Manager%TYPE,
      p_Email IN Administration.Email%TYPE
   );


   PROCEDURE Oda_Guncelle(
    p_Room_ID IN Room.Room_ID%TYPE,
    p_Surgery_ID IN Room.Surgery_ID%TYPE,
    p_Status IN Room.Status%TYPE,
    p_Department_Name IN Room.Department_Name%TYPE,
    p_Room_Type IN Room.Room_Type%TYPE
 );
 PROCEDURE Departman_Guncelle(
    p_Department_ID IN Department.Department_ID%TYPE,
    p_Department_Name IN Department.Department_Name%TYPE,
    p_Stuff IN Department.Stuff%TYPE,
    p_Department_Description IN Department.Department_Description%TYPE
 );
PROCEDURE Calisan_Guncelle(
    p_Employee_ID IN Employee.Employee_ID%TYPE,
    p_Department_ID IN Employee.Department_ID%TYPE,
    p_Phone IN Employee.Phone%TYPE,
    p_Start_Date IN Employee.Start_Date%TYPE,
    p_Email IN Employee.Email%TYPE,
    p_Address IN Employee.Address%TYPE,
    p_Full_Name IN Employee.Full_Name%TYPE
 );
PROCEDURE Hemşire_Guncelle (
    p_nurse_id IN Nurse.Nurse_ID%TYPE,
    p_nurse_name IN Nurse.Nurse_Name%TYPE
 );
  PROCEDURE Laboratuvar_Guncelle (
    p_lab_id IN Laboratory.Lab_ID%TYPE,
    p_patient_id IN Laboratory.Patient_ID%TYPE,
    p_test_date IN Laboratory.Test_Date%TYPE,
    p_reference_range IN Laboratory.Reference_Range%TYPE,
    p_lab_stuff_name IN Laboratory.Lab_Stuff_Name%TYPE,
    p_results IN Laboratory.Results%TYPE,
    p_test_name IN Laboratory.Test_Name%TYPE
 );
 PROCEDURE Ekipman_Guncelle(
    p_Equipment_ID IN Equipment.Equipment_ID%TYPE,
    p_Department_ID IN Equipment.Department_ID%TYPE,
    p_Equipment_Name IN Equipment.Equipment_Name%TYPE,
    p_Equipment_Quantity IN Equipment.Equipment_Quantity%TYPE,
    p_Requested_Date IN Equipment.Requested_Date%TYPE
 );

  PROCEDURE Doktor_Guncelle (
    p_doctor_id IN Doctor.Doctor_ID%TYPE,
    p_department_id IN Doctor.Department_ID%TYPE,
    p_age IN Doctor.Age%TYPE,
    p_full_name IN Doctor.Full_Name%TYPE,
    p_salary IN Doctor.Salary%TYPE,
    p_phone IN Doctor.Phone%TYPE,
    p_email IN Doctor.Email%TYPE,
    p_specialization IN Doctor.Specialization%TYPE
);

 PROCEDURE Teşhis_Guncelle (
    p_patient_id IN Diagnosis.Patient_ID%TYPE,
    p_medicine_id IN Diagnosis.Medicine_ID%TYPE,
    p_complaints IN Diagnosis.Complaints%TYPE,
    p_diagnosis_date IN Diagnosis.Diagnosis_Date%TYPE,
    p_age IN Diagnosis.Age%TYPE,
    p_results IN Diagnosis.Results%TYPE
);

PROCEDURE İlaç_Guncelle (
    p_medicine_id IN Medicine.Medicine_ID%TYPE,
    p_producer IN Medicine.Producer%TYPE,
    p_medicine_type IN Medicine.Medicine_Type%TYPE,
    p_warehouse_quantity IN Medicine.Warehouse_Quantity%TYPE,
    p_medicine_name IN Medicine.Medicine_Name%TYPE,
    p_product_date IN Medicine.Product_Date%TYPE
);

PROCEDURE Ameliyat_Guncelle (
    p_surgery_id IN Surgery.Surgery_ID%TYPE,
    p_patient_id IN Surgery.Patient_ID%TYPE,
    p_surgery_date IN Surgery.Surgery_Date%TYPE,
    p_department_name IN Surgery.Department_Name%TYPE
);

PROCEDURE Çalışma_Guncelle (
    p_nurse_id IN Work.Nurse_ID%TYPE,
    p_department_id IN Work.Department_ID%TYPE
);

 PROCEDURE Yönetim_Guncelle (
    p_nurse_id IN Governs.Nurse_ID%TYPE,
    p_room_id IN Governs.Room_ID%TYPE
);

PROCEDURE Randevu_Guncelle (
    p_patient_id IN Appoints.Patient_ID%TYPE,
    p_doctor_id IN Appoints.Doctor_ID%TYPE,
    p_appointment_date IN Appoints.Appointment_Date%TYPE,
    p_appointment_time IN Appoints.Appointment_Time%TYPE
);
```

```
END Veri_Guncelleme_Package;
/

CREATE OR REPLACE PACKAGE BODY Veri_Guncelleme_Package AS
  PROCEDURE Hasta_Guncelle (
    p_patient_id IN Patient.Patient_ID%TYPE,
    p_phone IN Patient.Phone%TYPE,
    p_full_name IN Patient.Full_Name%TYPE,
    p_gender IN Patient.Gender%TYPE,
    p_blood_type IN Patient.Blood_Type%TYPE,
    p_address IN Patient.Address%TYPE,
    p_emergency_person IN Patient.Emergency_Person%TYPE,
    p_age IN Patient.Age%TYPE
  ) IS
  BEGIN
    UPDATE Patient
    SET Phone = p_phone,
      Full_Name = p_full_name,
      Gender = p_gender,
      Blood_Type = p_blood_type,
      Address = p_address,
      Emergency_Person = p_emergency_person,
      Age = p_age
    WHERE Patient_ID = p_patient_id;
    COMMIT;
  END Hasta_Guncelle;

 PROCEDURE Ameliyat_Guncelle (
    p_surgery_id IN Surgery.Surgery_ID%TYPE,
    p_patient_id IN Surgery.Patient_ID%TYPE,
    p_surgery_date IN Surgery.Surgery_Date%TYPE
  ) IS
  BEGIN
    UPDATE Surgery
    SET Patient_ID = p_patient_id,
      Surgery_Date = p_surgery_date
    WHERE Surgery_ID = p_surgery_id;
    COMMIT;
  END Ameliyat_Guncelle;

  PROCEDURE Doktor_Guncelle (
    p_doctor_id IN Doctor.Doctor_ID%TYPE,
    p_department_id IN Doctor.Department_ID%TYPE,
    p_age IN Doctor.Age%TYPE,
    p_full_name IN Doctor.Full_Name%TYPE,
    p_salary IN Doctor.Salary%TYPE,
   p_phone IN Doctor.Phone%TYPE,
   p_email IN Doctor.Email%TYPE,
   p_specialization IN Doctor.Specialization%TYPE
) IS
BEGIN
   UPDATE Doctor
   SET Department_ID = p_department_id,
      Age = p_age,
      Full_Name = p_full_name,
      Salary = p_salary,
      Phone = p_phone,
      Email = p_email,
      Specialization = p_specialization
   WHERE Doctor_ID = p_doctor_id;
   COMMIT;
END Doktor_Guncelle;

PROCEDURE Teşhis_Guncelle (
   p_patient_id IN Diagnosis.Patient_ID%TYPE,
   p_medicine_id IN Diagnosis.Medicine_ID%TYPE,
   p_complaints IN Diagnosis.Complaints%TYPE,
   p_diagnosis_date IN Diagnosis.Diagnosis_Date%TYPE,
   p_age IN Diagnosis.Age%TYPE,
   p_results IN Diagnosis.Results%TYPE
) IS
BEGIN
   UPDATE Diagnosis
   SET Medicine_ID = p_medicine_id,
      Complaints = p_complaints,
      Diagnosis_Date = p_diagnosis_date,
      Age = p_age,
      Results = p_results
   WHERE Patient_ID = p_patient_id;
   COMMIT;
END Teşhis_Guncelle;

 PROCEDURE Kayit_Guncelle(
   p_Patient_ID IN Registration.Patient_ID%TYPE,
   p_Full_Name IN Registration.Full_Name%TYPE,
   p_Assurance IN Registration.Assurance%TYPE,
   p_Policlinic IN Registration.Policlinic%TYPE,
   p_Age IN Registration.Age%TYPE,
   p_R_Date IN Registration.R_Date%TYPE
) IS
BEGIN
   UPDATE Registration
   SET Full_Name = p_Full_Name,
        Assurance = p_Assurance,
        Policlinic = p_Policlinic,
        Age = p_Age,
        R_Date = p_R_Date
    WHERE Patient_ID = p_Patient_ID;
    COMMIT;
 END Kayit_Guncelle;

PROCEDURE Hastane_Guncelle(
    p_Hospital_ID IN Hospital.Hospital_ID%TYPE,
    p_Hospital_Name IN Hospital.Hospital_Name%TYPE,
    p_Capacity IN Hospital.Capacity%TYPE,
    p_Phone IN Hospital.Phone%TYPE,
    p_Total_Employee IN Hospital.Total_Employee%TYPE,
    p_Address IN Hospital.Address%TYPE,
    p_Founded_Date IN Hospital.Founded_Date%TYPE,
    p_Email IN Hospital.Email%TYPE
 ) IS
 BEGIN
    UPDATE Hospital
    SET Hospital_Name = p_Hospital_Name,
        Capacity = p_Capacity,
        Phone = p_Phone,
        Total_Employee = p_Total_Employee,
        Address = p_Address,
        Founded_Date = p_Founded_Date,
        Email = p_Email
    WHERE Hospital_ID = p_Hospital_ID;
    COMMIT;
 END Hastane_Guncelle;

 PROCEDURE Oda_Guncelle(
    p_Room_ID IN Room.Room_ID%TYPE,
    p_Surgery_ID IN Room.Surgery_ID%TYPE,
    p_Status IN Room.Status%TYPE,
    p_Department_Name IN Room.Department_Name%TYPE,
    p_Room_Type IN Room.Room_Type%TYPE
 ) IS
 BEGIN
    UPDATE Room
    SET Surgery_ID = p_Surgery_ID,
        Status = p_Status,
        Department_Name = p_Department_Name,
        Room_Type = p_Room_Type
    WHERE Room_ID = p_Room_ID;
    COMMIT;
 END Oda_Guncelle;
```

```
PROCEDURE Yonetim_Guncelle(
   p_Hospital_ID IN Administration.Hospital_ID%TYPE,
   p_Phone IN Administration.Phone%TYPE,
   p_Head_Manager IN Administration.Head_Manager%TYPE,
   p_Email IN Administration.Email%TYPE
) IS
BEGIN
   UPDATE Administration
   SET Phone = p_Phone,
      Head_Manager = p_Head_Manager,
      Email = p_Email
   WHERE Hospital_ID = p_Hospital_ID;
   COMMIT;
END Yonetim_Guncelle;

PROCEDURE Departman_Guncelle(
   p_Department_ID IN Department.Department_ID%TYPE,
   p_Department_Name IN Department.Department_Name%TYPE,
   p_Stuff IN Department.Stuff%TYPE,
   p_Department_Description IN Department.Department_Description%TYPE
) IS
BEGIN
   UPDATE Department
   SET Department_Name = p_Department_Name,
      Stuff = p_Stuff,
      Department_Description = p_Department_Description
   WHERE Department_ID = p_Department_ID;
   COMMIT;
END Departman_Guncelle;

 PROCEDURE Calisan_Guncelle(
   p_Employee_ID IN Employee.Employee_ID%TYPE,
   p_Department_ID IN Employee.Department_ID%TYPE,
   p_Phone IN Employee.Phone%TYPE,
   p_Start_Date IN Employee.Start_Date%TYPE,
   p_Email IN Employee.Email%TYPE,
   p_Address IN Employee.Address%TYPE,
   p_Full_Name IN Employee.Full_Name%TYPE
) IS
BEGIN
   UPDATE Employee
   SET Department_ID = p_Department_ID,
      Phone = p_Phone,
      Start_Date = p_Start_Date,
      Email = p_Email,
      Address = p_Address,
        Full_Name = p_Full_Name
    WHERE Employee_ID = p_Employee_ID;
    COMMIT;
 END Calisan_Guncelle;
 PROCEDURE Hemşire_Guncelle (
    p_nurse_id IN Nurse.Nurse_ID%TYPE,
    p_nurse_name IN Nurse.Nurse_Name%TYPE
 ) IS
 BEGIN
    UPDATE Nurse
    SET Nurse_Name = p_nurse_name
    WHERE Nurse_ID = p_nurse_id;
    COMMIT;
 END Hemşire_Guncelle;

PROCEDURE Laboratuvar_Guncelle (
    p_lab_id IN Laboratory.Lab_ID%TYPE,
    p_patient_id IN Laboratory.Patient_ID%TYPE,
    p_test_date IN Laboratory.Test_Date%TYPE,
    p_reference_range IN Laboratory.Reference_Range%TYPE,
    p_lab_stuff_name IN Laboratory.Lab_Stuff_Name%TYPE,
    p_results IN Laboratory.Results%TYPE,
    p_test_name IN Laboratory.Test_Name%TYPE
 ) IS
 BEGIN
    UPDATE Laboratory
    SET Patient_ID = p_patient_id,
        Test_Date = p_test_date,
        Reference_Range = p_reference_range,
        Lab_Stuff_Name = p_lab_stuff_name,
        Results = p_results,
        Test_Name = p_test_name
    WHERE Lab_ID = p_lab_id;
    COMMIT;
 END Laboratuvar_Guncelle;

 PROCEDURE Ekipman_Guncelle(
    p_Equipment_ID IN Equipment.Equipment_ID%TYPE,
    p_Department_ID IN Equipment.Department_ID%TYPE,
    p_Equipment_Name IN Equipment.Equipment_Name%TYPE,
    p_Equipment_Quantity IN Equipment.Equipment_Quantity%TYPE,
    p_Requested_Date IN Equipment.Requested_Date%TYPE
 ) IS
 BEGIN
    UPDATE Equipment
    SET Department_ID = p_Department_ID,
        Equipment_Name = p_Equipment_Name,
        Equipment_Quantity = p_Equipment_Quantity,
        Requested_Date = p_Requested_Date
    WHERE Equipment_ID = p_Equipment_ID;
    COMMIT;
END Ekipman_Guncelle;

PROCEDURE Hasta_Guncelle (
    p_patient_id IN Patient.Patient_ID%TYPE,
    p_phone IN Patient.Phone%TYPE,
    p_full_name IN Patient.Full_Name%TYPE,
    p_gender IN Patient.Gender%TYPE,
    p_blood_type IN Patient.Blood_Type%TYPE,
    p_address IN Patient.Address%TYPE,
    p_emergency_person IN Patient.Emergency_Person%TYPE,
    p_age IN Patient.Age%TYPE
) IS
BEGIN
    UPDATE Patient
    SET Phone = p_phone,
        Full_Name = p_full_name,
        Gender = p_gender,
        Blood_Type = p_blood_type,
        Address = p_address,
        Emergency_Person = p_emergency_person,
        Age = p_age
    WHERE Patient_ID = p_patient_id;

    COMMIT;
END Hasta_Guncelle;

PROCEDURE Doktor_Guncelle (
    p_doctor_id IN Doctor.Doctor_ID%TYPE,
    p_department_id IN Doctor.Department_ID%TYPE,
    p_age IN Doctor.Age%TYPE,
    p_full_name IN Doctor.Full_Name%TYPE,
    p_salary IN Doctor.Salary%TYPE,
    p_phone IN Doctor.Phone%TYPE,
    p_email IN Doctor.Email%TYPE,
    p_specialization IN Doctor.Specialization%TYPE
) IS
BEGIN
    UPDATE Doctor
    SET Department_ID = p_department_id,
        Age = p_age,
        Full_Name = p_full_name,
        Salary = p_salary,
        Phone = p_phone,
        Email = p_email,
        Specialization = p_specialization
    WHERE Doctor_ID = p_doctor_id;

    COMMIT;
END Doktor_Guncelle;

PROCEDURE Teşhis_Guncelle (
    p_patient_id IN Diagnosis.Patient_ID%TYPE,
    p_medicine_id IN Diagnosis.Medicine_ID%TYPE,
    p_complaints IN Diagnosis.Complaints%TYPE,
    p_diagnosis_date IN Diagnosis.Diagnosis_Date%TYPE,
    p_age IN Diagnosis.Age%TYPE,
    p_results IN Diagnosis.Results%TYPE
) IS
BEGIN
    UPDATE Diagnosis
    SET Patient_ID = p_patient_id,
        Medicine_ID = p_medicine_id,
        Complaints = p_complaints,
        Diagnosis_Date = p_diagnosis_date,
        Age = p_age,
        Results = p_results
    WHERE Diagnosis_ID = p_diagnosis_id;

    COMMIT;
END Teşhis_Guncelle;

 PROCEDURE İlaç_Guncelle (
    p_medicine_id IN Medicine.Medicine_ID%TYPE,
    p_producer IN Medicine.Producer%TYPE,
    p_medicine_type IN Medicine.Medicine_Type%TYPE,
    p_warehouse_quantity IN Medicine.Warehouse_Quantity%TYPE,
    p_medicine_name IN Medicine.Medicine_Name%TYPE,
    p_product_date IN Medicine.Product_Date%TYPE
) IS
BEGIN
    UPDATE Medicine
    SET Producer = p_producer,
        Medicine_Type = p_medicine_type,
        Warehouse_Quantity = p_warehouse_quantity,
        Medicine_Name = p_medicine_name,
        Product_Date = p_product_date
    WHERE Medicine_ID = p_medicine_id;

    COMMIT;
END İlaç_Guncelle;
```

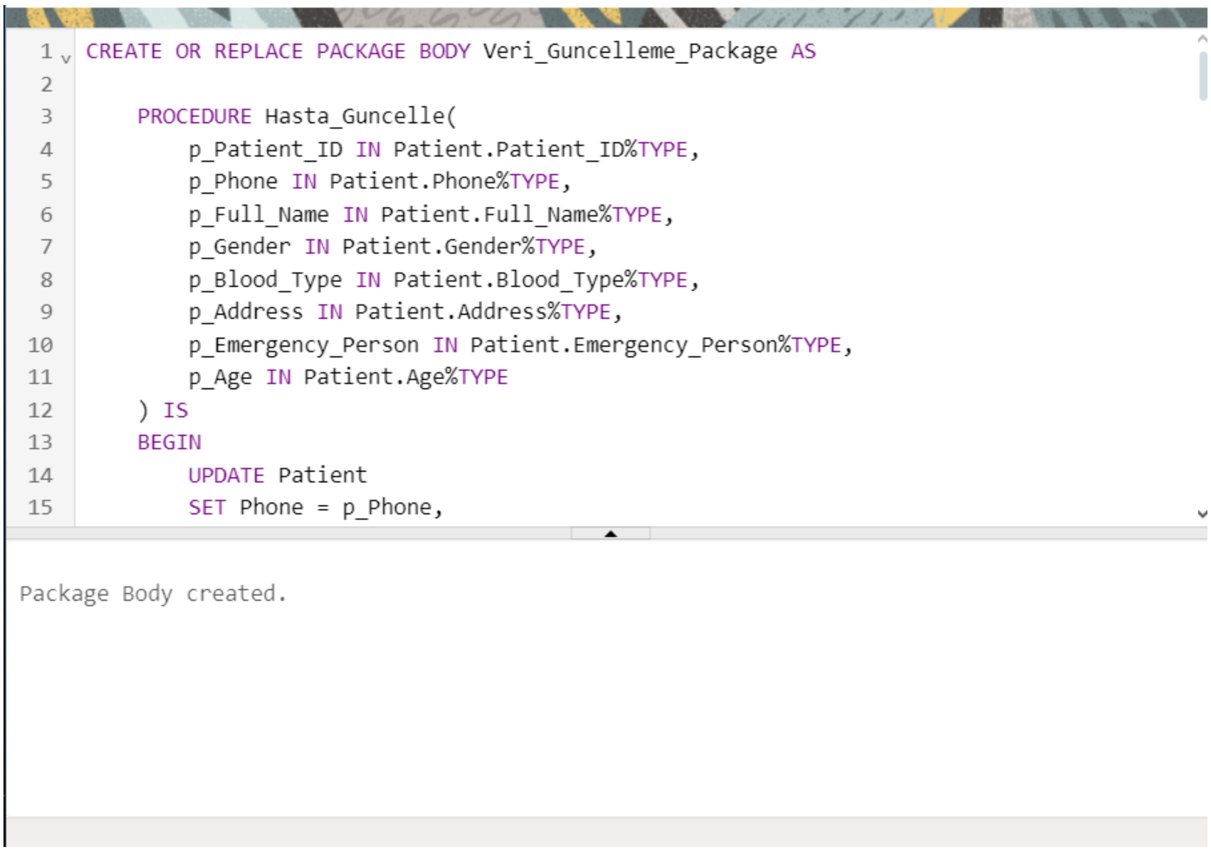
```
 PROCEDURE Ameliyat_Guncelle (
    p_surgery_id IN Surgery.Surgery_ID%TYPE,
    p_patient_id IN Surgery.Patient_ID%TYPE,
    p_surgery_date IN Surgery.Surgery_Date%TYPE,
    p_department_name IN Surgery.Department_Name%TYPE
) IS
BEGIN
    UPDATE Surgery
    SET Patient_ID = p_patient_id,
        Surgery_Date = p_surgery_date,
        Department_Name = p_department_name
    WHERE Surgery_ID = p_surgery_id;

    COMMIT;
END Ameliyat_Guncelle;

 PROCEDURE Randevu_Guncelle (
    p_patient_id IN Appoints.Patient_ID%TYPE,
    p_doctor_id IN Appoints.Doctor_ID%TYPE,
    p_appointment_date IN Appoints.Appointment_Date%TYPE,
    p_appointment_time IN Appoints.Appointment_Time%TYPE
) IS
BEGIN
    UPDATE Appoints
    SET Appointment_Date = p_appointment_date,
        Appointment_Time = p_appointment_time
    WHERE Patient_ID = p_patient_id
      AND Doctor_ID = p_doctor_id;

    COMMIT;
END Randevu_Guncelle;

PROCEDURE Çalışma_Guncelle (
    p_nurse_id IN Work.Nurse_ID%TYPE,
    p_department_id IN Work.Department_ID%TYPE
) IS
BEGIN
    UPDATE Work
    SET Department_ID = p_department_id
    WHERE Nurse_ID = p_nurse_id;

    COMMIT;
END Çalışma_Guncelle;

PROCEDURE Yönetim_Guncelle (
    p_nurse_id IN Governs.Nurse_ID%TYPE,
        p_room_id IN Governs.Room_ID%TYPE
    ) IS
    BEGIN
        UPDATE Governs
        SET Room_ID = p_room_id
        WHERE Nurse_ID = p_nurse_id;

        COMMIT;
    END Yönetim_Guncelle;

END Veri_Guncelleme_Package;
/
```

ÇIKTI:



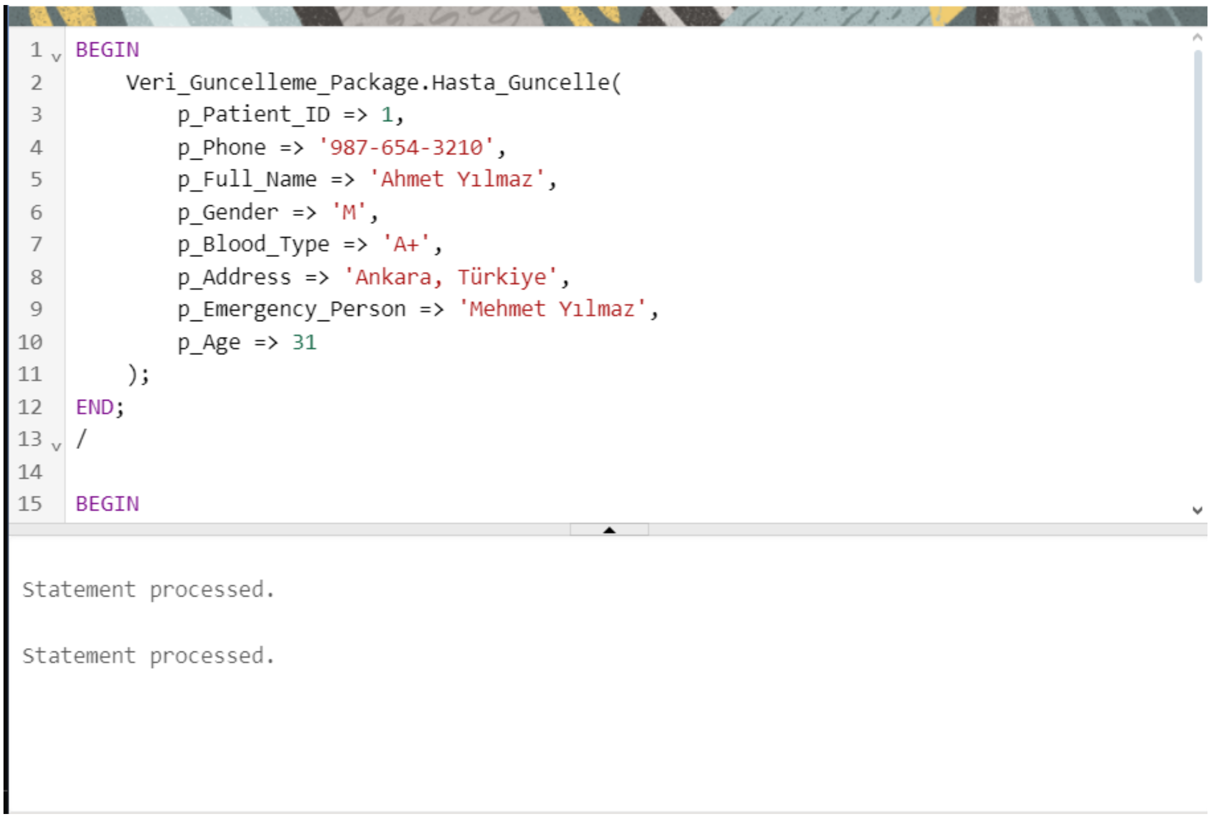
```
CREATE OR REPLACE PACKAGE Veri_Guncelleme_Package AS
    PROCEDURE Hasta_Guncelle(
        p_Patient_ID IN Patient.Patient_ID%TYPE,
        p_Phone IN Patient.Phone%TYPE,
        p_Full_Name IN Patient.Full_Name%TYPE,
        p_Gender IN Patient.Gender%TYPE,
        p_Blood_Type IN Patient.Blood_Type%TYPE,
        p_Address IN Patient.Address%TYPE,
        p_Emergency_Person IN Patient.Emergency_Person%TYPE,
        p_Age IN Patient.Age%TYPE
    );

    PROCEDURE Doktor_Guncelle(
        p_Doctor_ID IN Doctor.Doctor_ID%TYPE,
        p_Patient_ID IN Doctor.Patient_ID%TYPE,
```

```
Package created.
```



```
CREATE OR REPLACE PACKAGE BODY Veri_Guncelleme_Package AS

    PROCEDURE Hasta_Guncelle(
        p_Patient_ID IN Patient.Patient_ID%TYPE,
        p_Phone IN Patient.Phone%TYPE,
        p_Full_Name IN Patient.Full_Name%TYPE,
        p_Gender IN Patient.Gender%TYPE,
        p_Blood_Type IN Patient.Blood_Type%TYPE,
        p_Address IN Patient.Address%TYPE,
        p_Emergency_Person IN Patient.Emergency_Person%TYPE,
        p_Age IN Patient.Age%TYPE
    ) IS
    BEGIN
        UPDATE Patient
        SET Phone = p_Phone,
```

```
Package Body created.
```



```
BEGIN
    Veri_Guncelleme_Package.Hasta_Guncelle(
        p_Patient_ID => 1,
        p_Phone => '987-654-3210',
        p_Full_Name => 'Ahmet Yılmaz',
        p_Gender => 'M',
        p_Blood_Type => 'A+',
        p_Address => 'Ankara, Türkiye',
        p_Emergency_Person => 'Mehmet Yılmaz',
        p_Age => 31
    );
END;
/

BEGIN
```

```
Statement processed.

Statement processed.
```

5)PROJENIZIN VERI SILME IŞLEMINI BIR PL/SQL PAKETTEN ÇAĞRILABILEN BIR KOD ARACILIĞI(PROSEDÜR VEYA FONKSIYON) ILE GERÇEKLEŞTIRINIZ.

```
CREATE OR REPLACE PACKAGE Veri_Silme_Package AS
   PROCEDURE Hasta_Sil(p_Patient_ID IN Patient.Patient_ID%TYPE);

   PROCEDURE Doktor_Sil(p_Doctor_ID IN Doctor.Doctor_ID%TYPE);

   PROCEDURE Kayit_Sil(p_Patient_ID IN Registration.Patient_ID%TYPE);

   PROCEDURE Hastane_Sil(p_Hospital_ID IN Hospital.Hospital_ID%TYPE);

   PROCEDURE Yonetim_Sil(p_Hospital_ID IN Administration.Hospital_ID%TYPE);

   PROCEDURE Teshis_Sil(p_Doctor_ID IN Diagnosis.Doctor_ID%TYPE, p_Medicine_ID IN
Diagnosis.Medicine_ID%TYPE);

   PROCEDURE Ilac_Sil(p_Medicine_ID IN Medicine.Medicine_ID%TYPE);

   PROCEDURE Oda_Sil(p_Room_ID IN Room.Room_ID%TYPE);

   PROCEDURE Hemsire_Sil(p_Nurse_ID IN Nurse.Nurse_ID%TYPE);

   PROCEDURE Ameliyat_Sil(p_Surgery_ID IN Surgery.Surgery_ID%TYPE);

   PROCEDURE Departman_Sil(p_Department_ID IN Department.Department_ID%TYPE);

   PROCEDURE Calisan_Sil(p_Employee_ID IN Employee.Employee_ID%TYPE);

   PROCEDURE Laboratuvar_Sil(p_Lab_ID IN Laboratory.Lab_ID%TYPE);

   PROCEDURE Ekipman_Sil(p_Equipment_ID IN Equipment.Equipment_ID%TYPE);

END Veri_Silme_Package;
/

CREATE OR REPLACE PACKAGE Veri_Silme_Paketi AS
   -- Ameliyat tablosundan başlayarak veri silme işlemi
   PROCEDURE Ameliyat_Silme (p_surgery_id IN Surgery.Surgery_ID%TYPE);

   -- Randevu Atama tablosundan veri silme işlemi
   PROCEDURE Randevu_Silme (p_patient_id IN Appoints.Patient_ID%TYPE, p_doctor_id IN
Appoints.Doctor_ID%TYPE);
```

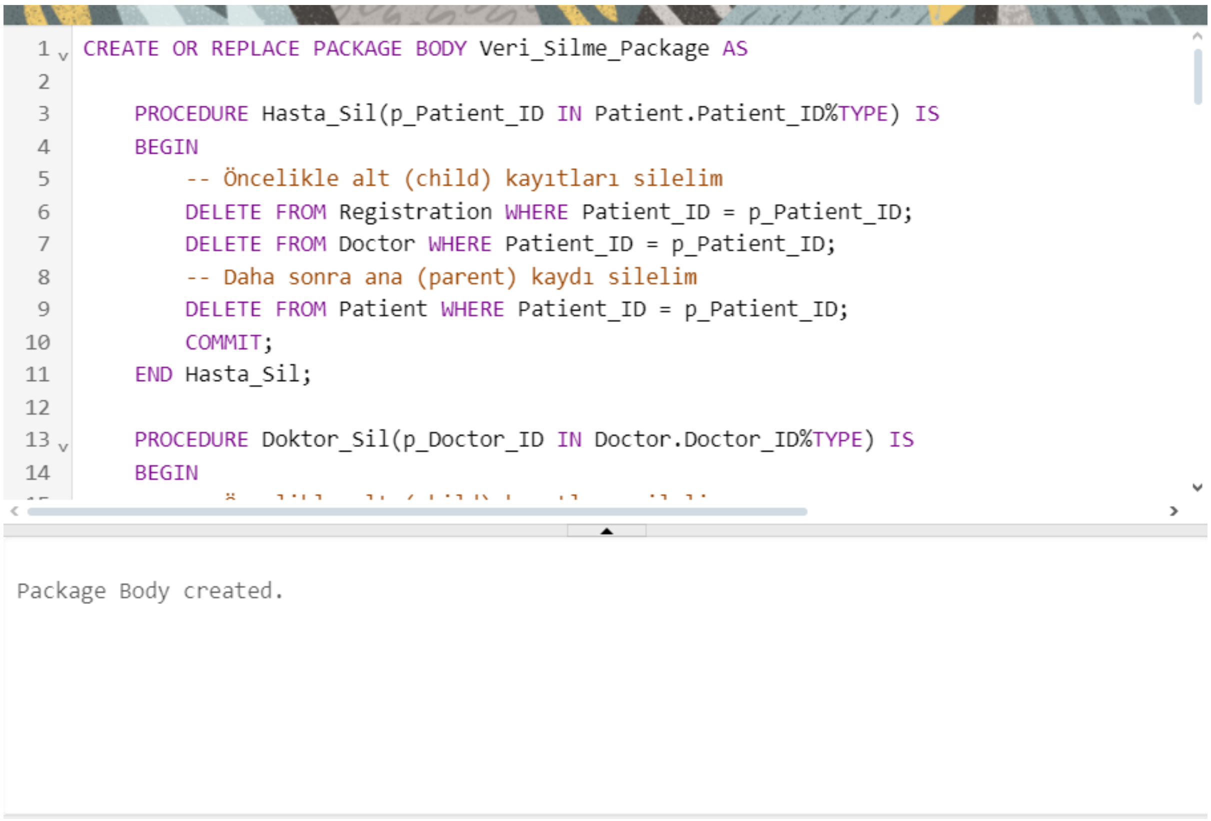
```
    -- Çalışma tablosundan veri silme işlemi
    PROCEDURE Çalışma_Silme (p_nurse_id IN Work.Nurse_ID%TYPE);

    -- Hemşire tablosundan veri silme işlemi
    PROCEDURE Hemşire_Silme (p_nurse_id IN Nurse.Nurse_ID%TYPE);

    -- Oda tablosundan veri silme işlemi
    PROCEDURE Oda_Silme (p_room_id IN Room.Room_ID%TYPE);

    -- Yönetim tablosundan veri silme işlemi
    PROCEDURE Yönetim_Silme (p_nurse_id IN Governs.Nurse_ID%TYPE, p_room_id IN
Governs.Room_ID%TYPE);

    -- Laboratuvar tablosundan veri silme işlemi
    PROCEDURE Laboratuvar_Silme (p_lab_id IN Laboratory.Lab_ID%TYPE);

    -- İlaç tablosundan veri silme işlemi
    PROCEDURE İlaç_Silme (p_medicine_id IN Medicine.Medicine_ID%TYPE);

    -- Teşhis tablosundan veri silme işlemi
    PROCEDURE Teşhis_Silme (p_diagnosis_id IN Diagnosis.Diagnosis_ID%TYPE);

    -- Doktor tablosundan veri silme işlemi
    PROCEDURE Doktor_Silme (p_doctor_id IN Doctor.Doctor_ID%TYPE);

    -- Departman tablosundan veri silme işlemi
    PROCEDURE Departman_Silme (p_department_id IN Department.Department_ID%TYPE);

    -- Çalışan tablosundan veri silme işlemi
    PROCEDURE Çalışan_Silme (p_employee_id IN Employee.Employee_ID%TYPE);

    -- Ekipman tablosundan veri silme işlemi
    PROCEDURE Ekipman_Silme (p_equipment_id IN Equipment.Equipment_ID%TYPE);

    -- Hasta tablosundan veri silme işlemi
    PROCEDURE Hasta_Silme (p_patient_id IN Patient.Patient_ID%TYPE);
END Veri_Silme_Paketi;
/

CREATE OR REPLACE PACKAGE BODY Veri_Silme_Package AS

    PROCEDURE Ameliyat_Sil(p_Surgery_ID IN Surgery.Surgery_ID%TYPE) IS
    BEGIN
        DELETE FROM Room WHERE Surgery_ID = p_Surgery_ID; -- İlgili odaları kullanımdan sil
        DELETE FROM Surgery WHERE Surgery_ID = p_Surgery_ID;
        COMMIT;
    END Ameliyat_Sil;
```

```
  PROCEDURE Calisan_Sil(p_Employee_ID IN Employee.Employee_ID%TYPE) IS
  BEGIN
    DELETE FROM Employee WHERE Employee_ID = p_Employee_ID;
    COMMIT;
  END Calisan_Sil;

  PROCEDURE Departman_Sil(p_Department_ID IN Department.Department_ID%TYPE) IS
  BEGIN
    -- Öncelikle alt (child) kayıtları silelim
    DELETE FROM Doctor WHERE Department_ID = p_Department_ID;
    DELETE FROM Employee WHERE Department_ID = p_Department_ID;
    DELETE FROM Nurse WHERE Department_ID = p_Department_ID;
    DELETE FROM Equipment WHERE Department_ID = p_Department_ID;
    DELETE FROM Room WHERE Department_Name = (SELECT Department_Name FROM
Department WHERE Department_ID = p_Department_ID);
    -- Daha sonra ana (parent) kaydı silelim
    DELETE FROM Department WHERE Department_ID = p_Department_ID;
    COMMIT;
  END Departman_Sil;

  PROCEDURE Doktor_Sil(p_Doctor_ID IN Doctor.Doctor_ID%TYPE) IS
  BEGIN
    -- Öncelikle alt (child) kayıtları silelim
    DELETE FROM Diagnosis WHERE Doctor_ID = p_Doctor_ID;
    DELETE FROM Laboratory WHERE Doctor_ID = p_Doctor_ID;
    -- Daha sonra ana (parent) kaydı silelim
    DELETE FROM Doctor WHERE Doctor_ID = p_Doctor_ID;
    COMMIT;
  END Doktor_Sil;

  PROCEDURE Ekipman_Sil(p_Equipment_ID IN Equipment.Equipment_ID%TYPE) IS
  BEGIN
    DELETE FROM Equipment WHERE Equipment_ID = p_Equipment_ID;
    COMMIT;
  END Ekipman_Sil;

  PROCEDURE Hasta_Sil(p_Patient_ID IN Patient.Patient_ID%TYPE) IS
  BEGIN
    DELETE FROM Registration WHERE Patient_ID = p_Patient_ID;
    DELETE FROM Appoints WHERE Patient_ID = p_Patient_ID;
    DELETE FROM Diagnosis WHERE Patient_ID = p_Patient_ID;
    -- Daha sonra ana (parent) kaydı silelim
    DELETE FROM Patient WHERE Patient_ID = p_Patient_ID;
    COMMIT;
  END Hasta_Sil;
```

```
   PROCEDURE Hastane_Sil(p_Hospital_ID IN Hospital.Hospital_ID%TYPE) IS
   BEGIN
      -- Öncelikle alt (child) kayıtları silelim
      DELETE FROM Administration WHERE Hospital_ID = p_Hospital_ID;
      -- Daha sonra ana (parent) kaydı silelim
      DELETE FROM Hospital WHERE Hospital_ID = p_Hospital_ID;
      COMMIT;
   END Hastane_Sil;

   PROCEDURE Hemsire_Sil(p_Nurse_ID IN Nurse.Nurse_ID%TYPE) IS
   BEGIN
      DELETE FROM Nurse WHERE Nurse_ID = p_Nurse_ID;
      COMMIT;
   END Hemsire_Sil;

   PROCEDURE Ilac_Sil(p_Medicine_ID IN Medicine.Medicine_ID%TYPE) IS
   BEGIN
      DELETE FROM Diagnosis WHERE Medicine_ID = p_Medicine_ID; -- İlaç kayıtlarını
kullanımdan sil
      DELETE FROM Medicine WHERE Medicine_ID = p_Medicine_ID;
      COMMIT;
   END Ilac_Sil;

   PROCEDURE Kayit_Sil(p_Patient_ID IN Registration.Patient_ID%TYPE) IS
   BEGIN
      DELETE FROM Registration WHERE Patient_ID = p_Patient_ID;
      COMMIT;
   END Kayit_Sil;

   PROCEDURE Laboratuvar_Sil(p_Lab_ID IN Laboratory.Lab_ID%TYPE) IS
   BEGIN
      DELETE FROM Laboratory WHERE Lab_ID = p_Lab_ID;
      COMMIT;
   END Laboratuvar_Sil;

   PROCEDURE Oda_Sil(p_Room_ID IN Room.Room_ID%TYPE) IS
   BEGIN
      DELETE FROM Room WHERE Room_ID = p_Room_ID;
      COMMIT;
   END Oda_Sil;

   PROCEDURE Teshis_Sil(p_Patient_ID IN Diagnosis.Patient_ID%TYPE, p_Medicine_ID IN
Diagnosis.Medicine_ID%TYPE) IS
   BEGIN
      DELETE FROM Diagnosis WHERE Patient_ID = p_Patient_ID AND Medicine_ID =
p_Medicine_ID;
      COMMIT;
```

```
  END Teshis_Sil;

  PROCEDURE Hareket_Sil(p_Nurse_ID IN Nurse.Nurse_ID%TYPE, p_Room_ID IN
Room.Room_ID%TYPE) IS
  BEGIN
    DELETE FROM Governs WHERE Nurse_ID = p_Nurse_ID AND Room_ID = p_Room_ID;
    DELETE FROM Assigns WHERE Nurse_ID = p_Nurse_ID AND Room_ID = p_Room_ID;
    DELETE FROM Appoints WHERE Patient_ID = p_Patient_ID;
    COMMIT;
  END Hareket_Sil;

END Veri_Silme_Package;
/

BEGIN
  Veri_Silme_Package.Hasta_Sil(p_Patient_ID => 1);
END;
/

BEGIN
  Veri_Silme_Package.Doktor_Sil(p_Doctor_ID => 1);
END;
/

BEGIN
  Veri_Silme_Package.Kayit_Sil(p_Patient_ID => 1);
END;
/

BEGIN
  Veri_Silme_Package.Hastane_Sil(p_Hospital_ID => 1);
END;
/

BEGIN
  Veri_Silme_Package.Teshis_Sil(p_Doctor_ID => 1, p_Medicine_ID => 1);
END;
```

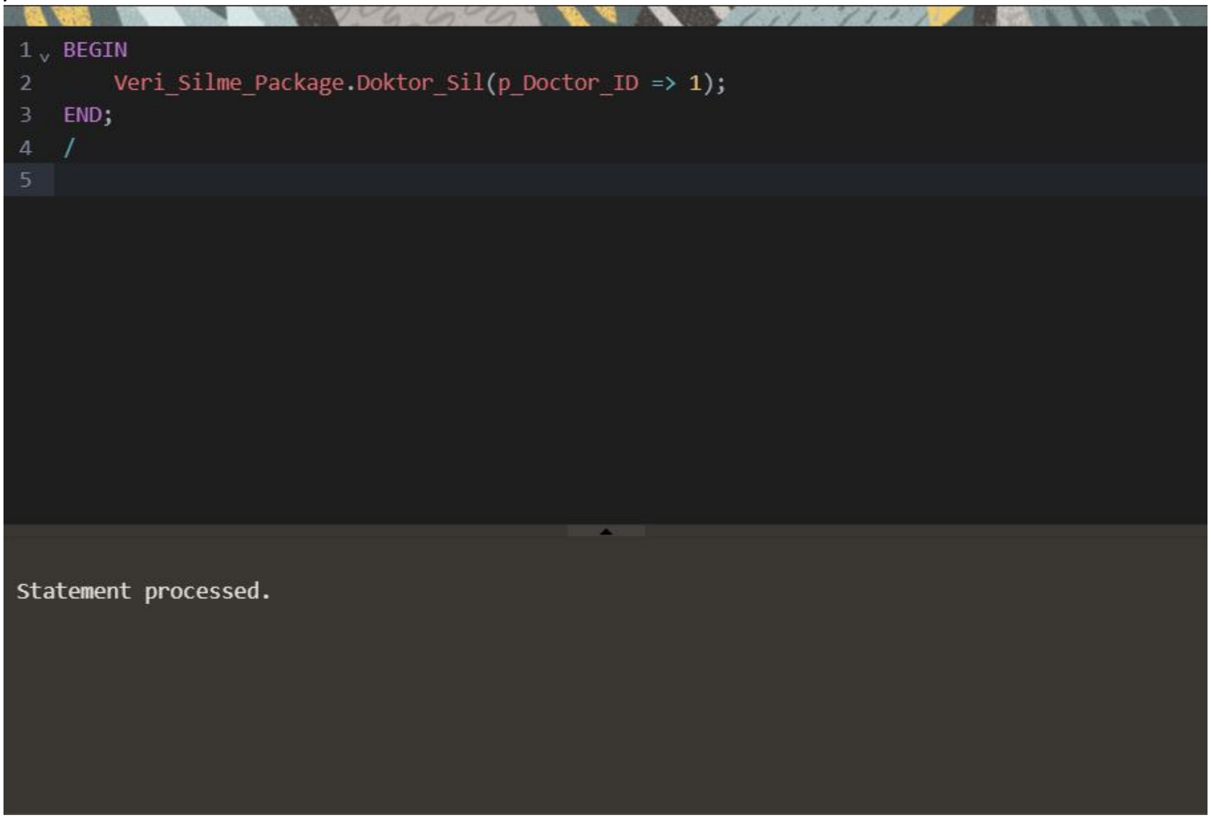
ÇIKTI:



```
CREATE OR REPLACE PACKAGE BODY Veri_Silme_Package AS

    PROCEDURE Hasta_Sil(p_Patient_ID IN Patient.Patient_ID%TYPE) IS
    BEGIN
        -- Öncelikle alt (child) kayıtları silelim
        DELETE FROM Registration WHERE Patient_ID = p_Patient_ID;
        DELETE FROM Doctor WHERE Patient_ID = p_Patient_ID;
        -- Daha sonra ana (parent) kaydı silelim
        DELETE FROM Patient WHERE Patient_ID = p_Patient_ID;
        COMMIT;
    END Hasta_Sil;

    PROCEDURE Doktor_Sil(p_Doctor_ID IN Doctor.Doctor_ID%TYPE) IS
    BEGIN
```

```
Package Body created.
```

/



```
BEGIN
    Veri_Silme_Package.Doktor_Sil(p_Doctor_ID => 1);
END;
/
```

```
Statement processed.
```

6)PL/SQL TETIK(TRIGGER) YARDIMI ILE BIR TANIM TABLOSUNA VERI GIRIŞI YAPILIRKEN, HAREKET TABLOSUNA DA VERI GIRIŞINI SAĞLAYINIZ.

Action_Log Tablosunun Oluşturulması
Öncelikle, Action_Log tablosunu oluşturalım. Bu tablo, gerçekleştirilen işlemleri kaydedecek.

```
CREATE TABLE Action_Log (
   Log_ID NUMBER PRIMARY KEY,
   Action_Type VARCHAR2(50),
   Table_Name VARCHAR2(50),
   Record_ID NUMBER,
   Action_Timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Sequence Oluşturma
Log_ID için bir sequence oluşturalım.

```
CREATE SEQUENCE LOG_SEQ
START WITH 1
INCREMENT BY 1
NOCACHE;
```

7)TRIGGER

Triggerin Tanımlanması

Patient tablosuna yeni bir kayıt eklendiğinde, Action_Log tablosuna bir kayıt ekleyecek triggeri oluşturalım.

```
CREATE OR REPLACE TRIGGER trg_after_insert_patient
AFTER INSERT ON Patient
FOR EACH ROW
BEGIN
   INSERT INTO Action_Log (Log_ID, Action_Type, Table_Name, Record_ID)
   VALUES (LOG_SEQ.NEXTVAL, 'INSERT', 'Patient', :NEW.Patient_ID);
END;
/
```

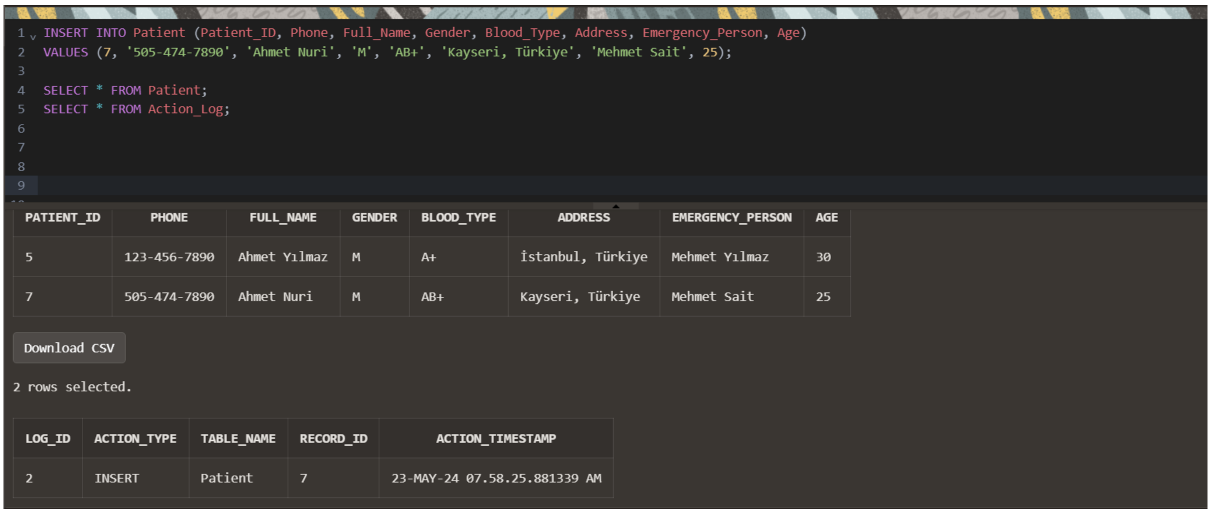
Örnek Veri Girişi

Şimdi, Patient tablosuna yeni bir kayıt eklediğimizde tetikleyicinin nasıl çalıştığını görelim.

Patient Tablosuna Veri Ekleme

INSERT INTO Patient (Patient_ID, Phone, Full_Name, Gender, Blood_Type, Address, Emergency_Person, Age)
VALUES (1, '123-456-7890', 'Ahmet Yılmaz', 'M', 'A+', 'İstanbul, Türkiye', 'Mehmet Yılmaz', 30);

ÇIKTI:



```
INSERT INTO Patient (Patient_ID, Phone, Full_Name, Gender, Blood_Type, Address, Emergency_Person, Age)
VALUES (7, '505-474-7890', 'Ahmet Nuri', 'M', 'AB+', 'Kayseri, Türkiye', 'Mehmet Sait', 25);

SELECT * FROM Patient;
SELECT * FROM Action_Log;
```

PATIENT_ID	PHONE	FULL_NAME	GENDER	BLOOD_TYPE	ADDRESS	EMERGENCY_PERSON	AGE
5	123-456-7890	Ahmet Yılmaz	M	A+	İstanbul, Türkiye	Mehmet Yılmaz	30
7	505-474-7890	Ahmet Nuri	M	AB+	Kayseri, Türkiye	Mehmet Sait	25

Download CSV

2 rows selected.

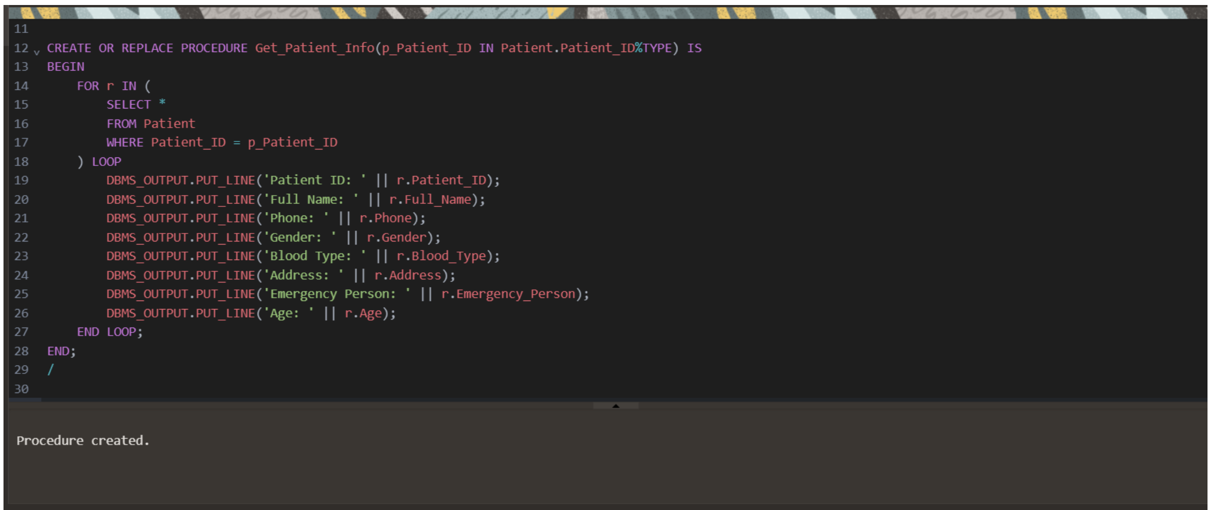
LOG_ID	ACTION_TYPE	TABLE_NAME	RECORD_ID	ACTION_TIMESTAMP
2	INSERT	Patient	7	23-MAY-24 07.58.25.881339 AM

8)BELLİ SORGULARLA RAPOR ALAN KOD ÖRNEKLERİ

1)Hasta ID sine göre hasta bilgilerini görüntüleme

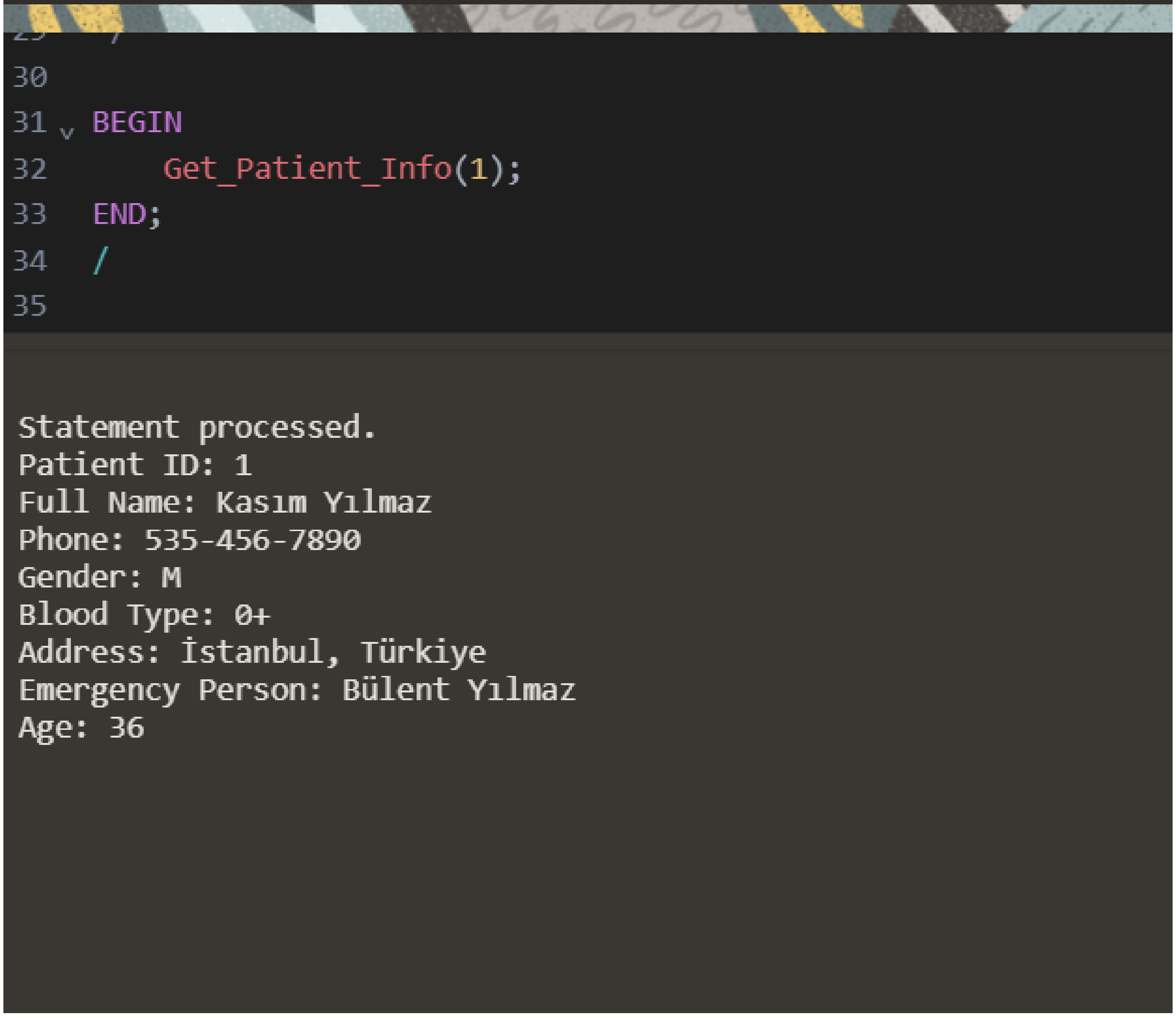
```
DECLARE
   v_patient_id Patient.Patient_ID%TYPE := 123; -- İstenen hasta ID'si
BEGIN
   FOR patient_info IN (SELECT * FROM Patient WHERE Patient_ID = v_patient_id) LOOP
      DBMS_OUTPUT.PUT_LINE('Patient ID: ' || patient_info.Patient_ID || ', Full Name: ' || patient_info.Full_Name || ', Gender: ' || patient_info.Gender);
   END LOOP;
END;
/
```

ÇIKTI:



```
CREATE OR REPLACE PROCEDURE Get_Patient_Info(p_Patient_ID IN Patient.Patient_ID%TYPE) IS
BEGIN
    FOR r IN (
        SELECT *
        FROM Patient
        WHERE Patient_ID = p_Patient_ID
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Patient ID: ' || r.Patient_ID);
        DBMS_OUTPUT.PUT_LINE('Full Name: ' || r.Full_Name);
        DBMS_OUTPUT.PUT_LINE('Phone: ' || r.Phone);
        DBMS_OUTPUT.PUT_LINE('Gender: ' || r.Gender);
        DBMS_OUTPUT.PUT_LINE('Blood Type: ' || r.Blood_Type);
        DBMS_OUTPUT.PUT_LINE('Address: ' || r.Address);
        DBMS_OUTPUT.PUT_LINE('Emergency Person: ' || r.Emergency_Person);
        DBMS_OUTPUT.PUT_LINE('Age: ' || r.Age);
    END LOOP;
END;
/
```

```
Procedure created.
```



```
BEGIN
    Get_Patient_Info(1);
END;
/
```

```
Statement processed.
Patient ID: 1
Full Name: Kasım Yılmaz
Phone: 535-456-7890
Gender: M
Blood Type: 0+
Address: İstanbul, Türkiye
Emergency Person: Bülent Yılmaz
Age: 36
```

2)Kan grubu A RH+ olan hastaları görüntüleme

```
-- Kan grubu A RH+ olan hastaları görüntülemek için prosedür
CREATE OR REPLACE PROCEDURE Get_Patients_By_Blood_Type(p_Blood_Type IN
Patient.Blood_Type%TYPE) IS
BEGIN
   FOR r IN (
      SELECT *
      FROM Patient
      WHERE Blood_Type = p_Blood_Type
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Patient ID: ' || r.Patient_ID);
      DBMS_OUTPUT.PUT_LINE('Full Name: ' || r.Full_Name);
      DBMS_OUTPUT.PUT_LINE('Phone: ' || r.Phone);
      DBMS_OUTPUT.PUT_LINE('Gender: ' || r.Gender);
      DBMS_OUTPUT.PUT_LINE('Blood Type: ' || r.Blood_Type);
      DBMS_OUTPUT.PUT_LINE('Address: ' || r.Address);
      DBMS_OUTPUT.PUT_LINE('Emergency Person: ' || r.Emergency_Person);
      DBMS_OUTPUT.PUT_LINE('Age: ' || r.Age);
   END LOOP;
END;
/
```

3)Cihaz sayısı 100'den fazla olan departmanları görüntülemek için prosedür

```
CREATE OR REPLACE PROCEDURE Get_Departments_With_More_Equipment AS
BEGIN
   FOR r IN (
      SELECT *
      FROM Department
      WHERE Equipment_Quantity > 100
   ) LOOP
      DBMS_OUTPUT.PUT_LINE('Department ID: ' || r.Department_ID);
      DBMS_OUTPUT.PUT_LINE('Department Name: ' || r.Department_Name);
      DBMS_OUTPUT.PUT_LINE('Stuff: ' || r.Stuff);
      DBMS_OUTPUT.PUT_LINE('Department Description: ' || r.Department_Description);
   END LOOP;
END;
/
```

4) 20.01.2022'den sonra üretilen ilaçları görüntülemek için prosedür

```
CREATE OR REPLACE PROCEDURE Get_Medicines_Produced_After_Date(p_Production_Date
IN DATE) IS
BEGIN
   FOR r IN (
      SELECT *
      FROM Medicine
      WHERE Product_Date > p_Production_Date
```

```
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Medicine ID: ' || r.Medicine_ID);
    DBMS_OUTPUT.PUT_LINE('Producer: ' || r.Producer);
    DBMS_OUTPUT.PUT_LINE('Medicine Type: ' || r.Medicine_Type);
    DBMS_OUTPUT.PUT_LINE('Warehouse Quantity: ' || r.Warehouse_Quantity);
    DBMS_OUTPUT.PUT_LINE('Medicine Name: ' || r.Medicine_Name);
    DBMS_OUTPUT.PUT_LINE('Product Date: ' || r.Product_Date);
  END LOOP;
END;
/
```

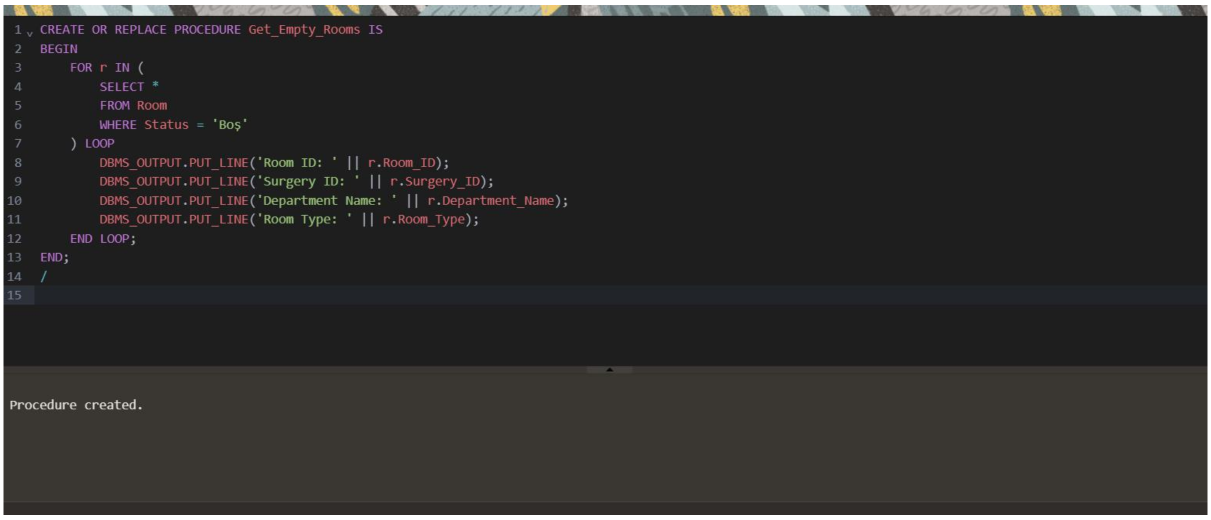
5) İlaçları 20.01.2022 den sonra üretilenler olarak görüntüleme

```
DECLARE
  v_production_date DATE := TO_DATE('2022-01-20', 'YYYY-MM-DD'); -- Üretim tarihi eşik
değeri
BEGIN
  FOR medicine_info IN (SELECT * FROM Medicine WHERE Product_Date >
v_production_date) LOOP
    DBMS_OUTPUT.PUT_LINE('Medicine ID: ' || medicine_info.Medicine_ID || ', Medicine
Name: ' || medicine_info.Medicine_Name || ', Production Date: ' ||
TO_CHAR(medicine_info.Product_Date, 'DD.MM.YYYY'));
    -- Diğer alanlar da buraya eklenebilir
  END LOOP;
END;
/
```

6) Oda durumunu gösteren kod sorgusu

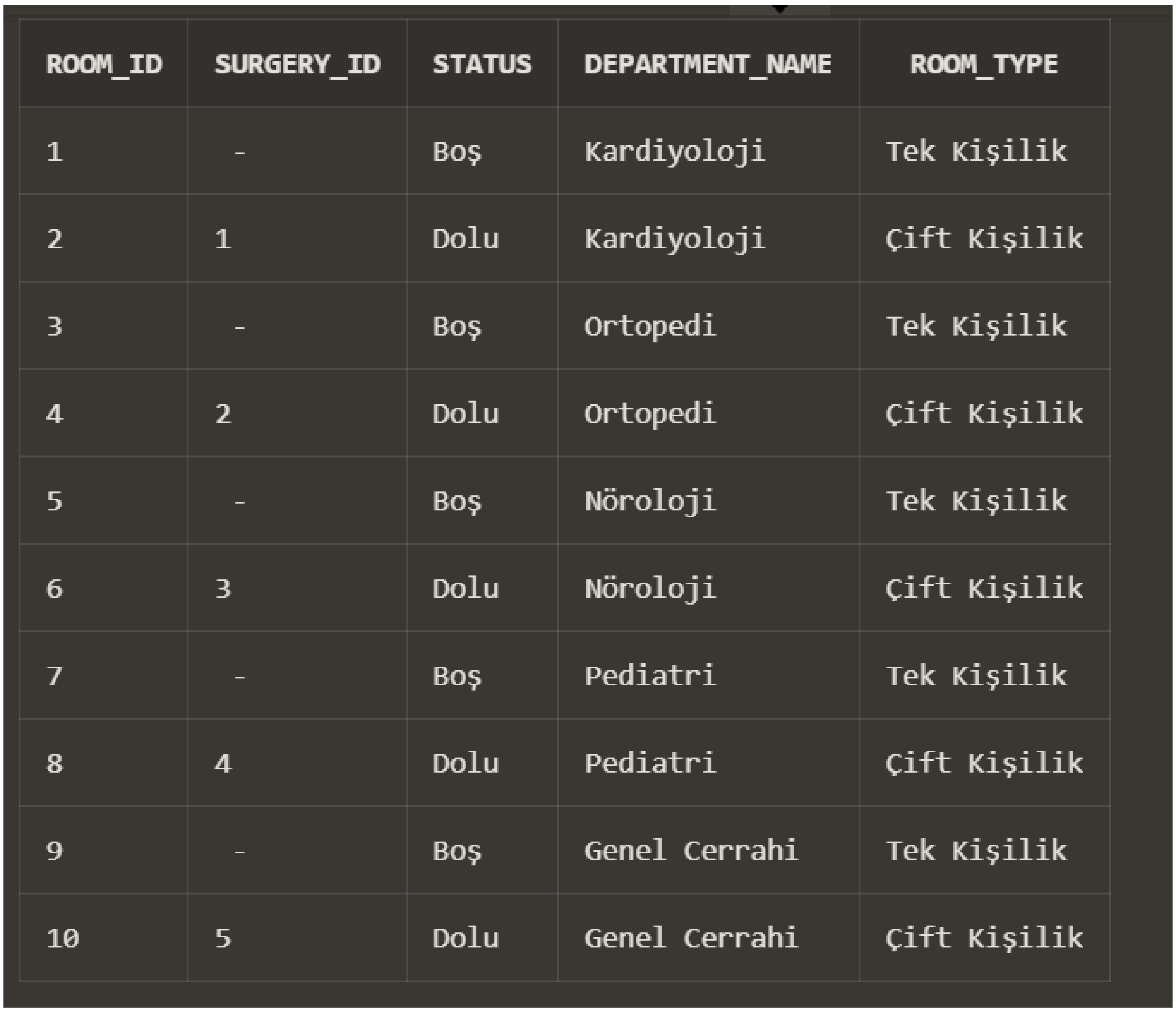
```
CREATE OR REPLACE PROCEDURE Get_Empty_Rooms IS
BEGIN
  FOR r IN (
    SELECT *
    FROM Room
    WHERE Status = 'Boş'
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Room ID: ' || r.Room_ID);
    DBMS_OUTPUT.PUT_LINE('Surgery ID: ' || r.Surgery_ID);
    DBMS_OUTPUT.PUT_LINE('Department Name: ' || r.Department_Name);
    DBMS_OUTPUT.PUT_LINE('Room Type: ' || r.Room_Type);
  END LOOP;
END;
/

BEGIN
  Get_Empty_Rooms;
END;
/
```

ÇIKTI:



```
CREATE OR REPLACE PROCEDURE Get_Empty_Rooms IS
BEGIN
    FOR r IN (
        SELECT *
        FROM Room
        WHERE Status = 'Boş'
    ) LOOP
        DBMS_OUTPUT.PUT_LINE('Room ID: ' || r.Room_ID);
        DBMS_OUTPUT.PUT_LINE('Surgery ID: ' || r.Surgery_ID);
        DBMS_OUTPUT.PUT_LINE('Department Name: ' || r.Department_Name);
        DBMS_OUTPUT.PUT_LINE('Room Type: ' || r.Room_Type);
    END LOOP;
END;
/
```

```
Procedure created.
```



ROOM_ID	SURGERY_ID	STATUS	DEPARTMENT_NAME	ROOM_TYPE
1	-	Boş	Kardiyoloji	Tek Kişilik
2	1	Dolu	Kardiyoloji	Çift Kişilik
3	-	Boş	Ortopedi	Tek Kişilik
4	2	Dolu	Ortopedi	Çift Kişilik
5	-	Boş	Nöroloji	Tek Kişilik
6	3	Dolu	Nöroloji	Çift Kişilik
7	-	Boş	Pediatri	Tek Kişilik
8	4	Dolu	Pediatri	Çift Kişilik
9	-	Boş	Genel Cerrahi	Tek Kişilik
10	5	Dolu	Genel Cerrahi	Çift Kişilik

9)ÜÇÜNCÜ NORMAL FORMUN GÖSTERİLMESİ

Normalizasyonun iki temel amacı vardır. Veri tabanında veri tekrarlarını ortadan kaldırmak ve veri tutarlılığını (doğruluğunu) artırmak.

Normalizasyon, veri tabanlarına seviyelerle (normal formlar) uygulanır. Bir veri tabanının bu normal formlardan herhangi birine uygun olduğunu söyleyebilmek için, söz konusu normal formun tüm kriterlerini eksiksiz yerine getiriyor olması şarttır.

Başarılı bir şekilde uygulandığında normalizasyon işlemi veri tabanının süratini büyük oranda artırır. Veri tabanının sabit diskteki boyutunu azaltır. Ayrıca veri tutarlılığını artırarak veri tekrarlarını engeller.

1NF (1. Normal Form)

Bir veri tabanının 1NF olabilmesi için aşağıdaki özellikleri karşılayabilmesi gerekir:

- Aynı tablo içinde tekrarlayan kolonlar bulunamaz,
- Her kolonda yalnızca bir değer bulunabilir.
- Her satır bir eşsiz anahtarla tanımlanmalıdır (Unique Key - Primary Key)

Tasarladığımız veritabanı örneğinde aynı tablo içinde tekrarlayan hiçbir attribute bulunmayıp, her sütun ayrı bir attribute değerine işaret eder ve tabloların sadece bir tane eşsiz primary key ‘i bulunmaktadır.

2NF (2. Normal Form)

Bir veri tabanının 2NF olabilmesi için aşağıdaki özellikleri karşılayabilmesi gerekir:

- Tablo 1NF olmalıdır,
- Anahtar olmayan değerler ile kompozit (bileşik) anahtarlar arasında kısmi (partial) bağımlılık durumu oluşmamalıdır. Kısmi bağımlılık durumu, anahtar olmayan herhangi bir değer kompozit bir anahtarın yalnızca bir kısmına bağıl ise oluşur. Herhangi bir veri alt kümesi birden çok satırda tekrarlanmamalıdır. Bu tür veri alt kümeleri için yeni tablolar oluşturulmalıdır.
- Ana tablo ile yeni tablolar arasında, dış anahtarlar (foreign key) kullanılarak ilişkiler tanımlanmalıdır.

  Veritabanımızda anahtar olmayan değerler ve kompozit değerler arasında partial dependency bulunmamaktadır. Ayrıca bir tablonun primary key’I diğer tabloda foreign key olarak tasarlanmış ya da many to many ilişkili tablolar bir hareket tablosunda tutulan foreign key ‘ler aracılığıyla ilişkili hale getirilmiştir.

3NF (3. Normal Form)

Bir veri tabanının 3NF olabilmesi için aşağıdaki özellikleri karşılayabilmesi gerekir:

- Veri tabanı 2NF olmalıdır,
- Anahtar olmayan hiç bir kolon bir diğerine (anahtar olmayan başka bir kolona) bağlı olmamalı ya da geçişken fonksiyonel bir bağımlılığı (transitional functional dependency) olmamalıdır. Başka bir deyişle her kolon eşsiz anahtara tam bağımlı olmak zorundadır.

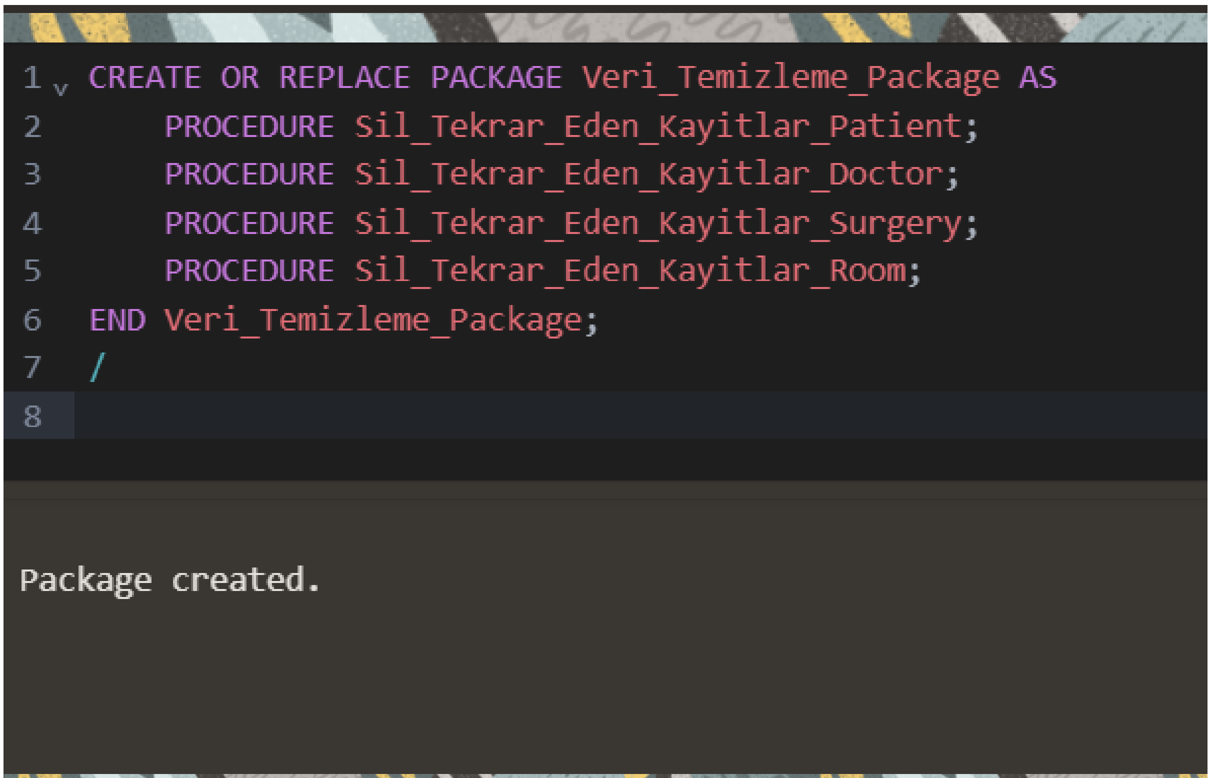
Tasarladığımız veritabanında her kolon yani ayrı bir attribute değerine denk gelen kolonlar,mutlaka bir primary key'e bağımlıdır. Bu sebeple veritabanımızın rahatlıkla 3.Normal Form kriterlerini karşıladığını söyleyebiliriz.

10) PL/SQL PAKETTEN ÇAĞRILABILEN PROSEDÜR VEYA FONKSIYON ARACILIĞI ILE TABLOLARDA TEKRAR EDEN KAYITLARI SILINIZ.

```
-- Veri Temizleme Paketi
CREATE OR REPLACE PACKAGE Veri_Temizleme_Package AS
   -- Patient tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Patient;

   -- Doctor tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Doctor;

   -- Room tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Room;

   -- Department tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Department;

   -- Medicine tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Medicine;

   -- Nurse tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Nurse;

   -- Equipment tablosu için
   PROCEDURE Sil_Tekrar_Eden_Kayitlar_Equipment;

END Veri_Temizleme_Package;
/

-- Veri Temizleme Package
```

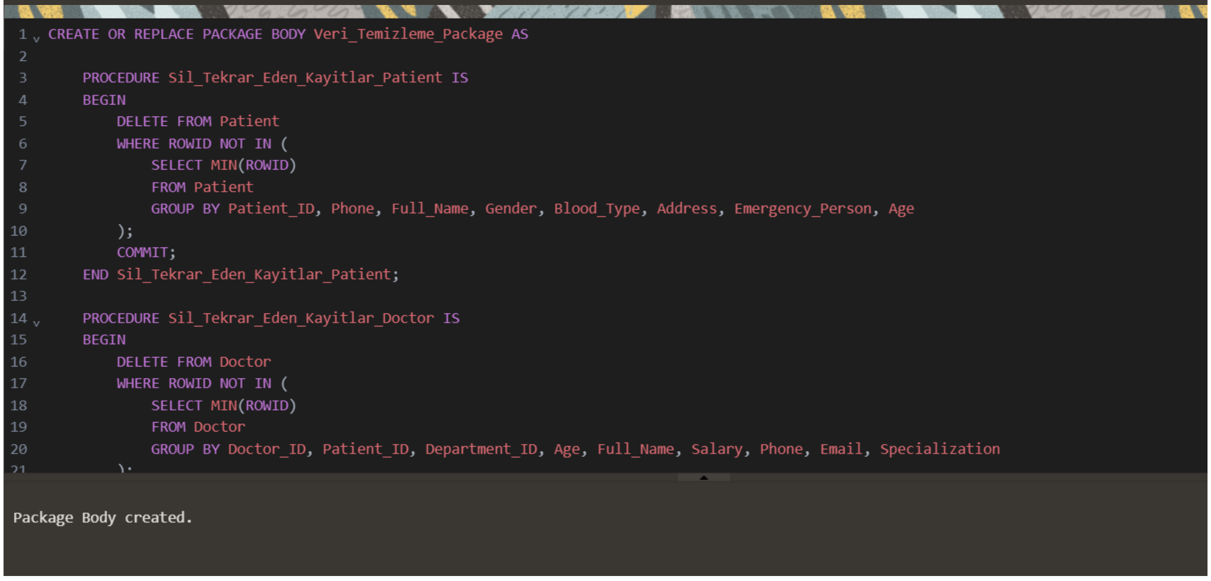
```
CREATE OR REPLACE PACKAGE BODY Veri_Temizleme_Package AS
  -- Patient tablosu için
  PROCEDURE Sil_Tekrar_Eden_Kayitlar_Patient IS
  BEGIN
    DELETE FROM Patient
    WHERE ROWID NOT IN (
      SELECT MIN(ROWID)
      FROM Patient
      GROUP BY Patient_ID
    );
    COMMIT;
  END Sil_Tekrar_Eden_Kayitlar_Patient;

  -- Doctor tablosu için
  PROCEDURE Sil_Tekrar_Eden_Kayitlar_Doctor IS
  BEGIN
    DELETE FROM Doctor
    WHERE ROWID NOT IN (
      SELECT MIN(ROWID)
      FROM Doctor
      GROUP BY Doctor_ID
    );
    COMMIT;
  END Sil_Tekrar_Eden_Kayitlar_Doctor;

  -- Room tablosu için
  PROCEDURE Sil_Tekrar_Eden_Kayitlar_Room IS
  BEGIN
    DELETE FROM Room
    WHERE ROWID NOT IN (
      SELECT MIN(ROWID)
      FROM Room
      GROUP BY Room_ID
    );
    COMMIT;
  END Sil_Tekrar_Eden_Kayitlar_Room;

  -- Department tablosu için
  PROCEDURE Sil_Tekrar_Eden_Kayitlar_Department IS
  BEGIN
    DELETE FROM Department
    WHERE ROWID NOT IN (
      SELECT MIN(ROWID)
      FROM Department
      GROUP BY Department_ID
    );
    COMMIT;
    END Sil_Tekrar_Eden_Kayitlar_Department;

    -- Medicine tablosu için
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Medicine IS
    BEGIN
        DELETE FROM Medicine
        WHERE ROWID NOT IN (
            SELECT MIN(ROWID)
            FROM Medicine
            GROUP BY Medicine_ID
        );
        COMMIT;
    END Sil_Tekrar_Eden_Kayitlar_Medicine;

    -- Nurse tablosu için
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Nurse IS
    BEGIN
        DELETE FROM Nurse
        WHERE ROWID NOT IN (
            SELECT MIN(ROWID)
            FROM Nurse
            GROUP BY Nurse_ID
        );
        COMMIT;
    END Sil_Tekrar_Eden_Kayitlar_Nurse;

    -- Equipment tablosu için
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Equipment IS
    BEGIN
        DELETE FROM Equipment
        WHERE ROWID NOT IN (
            SELECT MIN(ROWID)
            FROM Equipment
            GROUP BY Equipment_ID
        );
        COMMIT;
    END Sil_Tekrar_Eden_Kayitlar_Equipment;

END Veri_Temizleme_Package;
/
```

ÇIKTI: Prosedürü oluşturduk.



```
CREATE OR REPLACE PACKAGE Veri_Temizleme_Package AS
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Patient;
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Doctor;
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Surgery;
    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Room;
END Veri_Temizleme_Package;
/
```

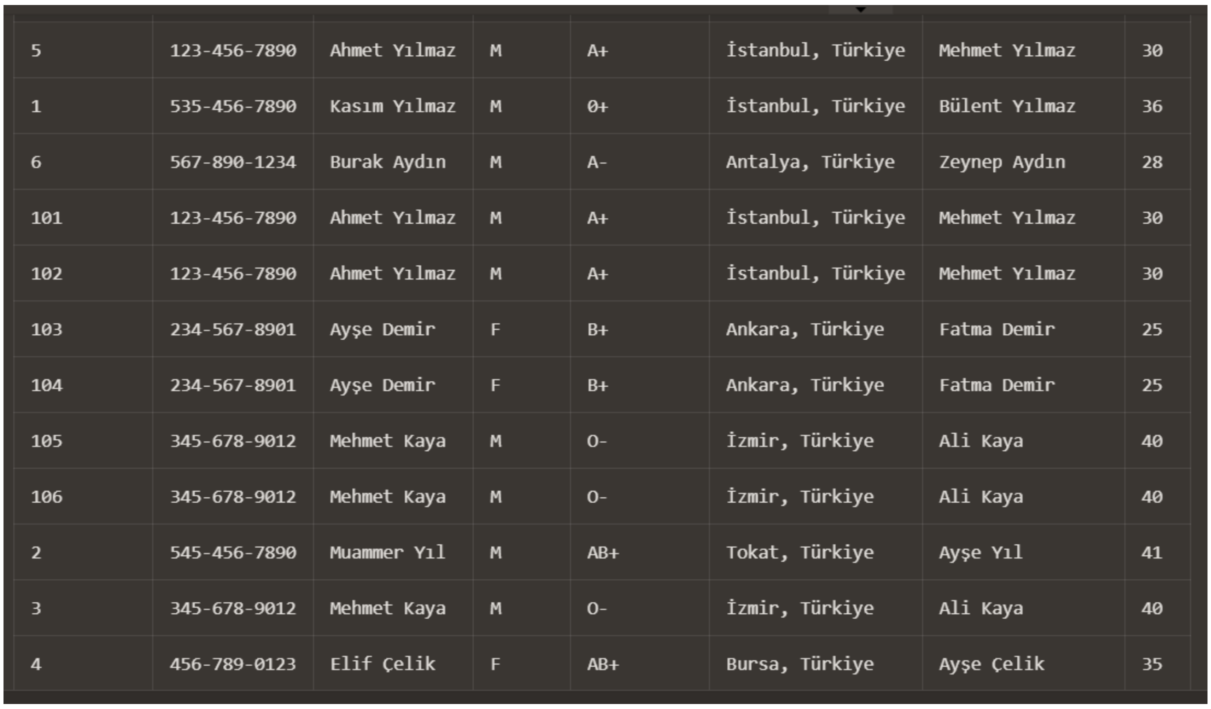
```
Package created.
```



```
CREATE OR REPLACE PACKAGE BODY Veri_Temizleme_Package AS

    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Patient IS
    BEGIN
        DELETE FROM Patient
        WHERE ROWID NOT IN (
            SELECT MIN(ROWID)
            FROM Patient
            GROUP BY Patient_ID, Phone, Full_Name, Gender, Blood_Type, Address, Emergency_Person, Age
        );
        COMMIT;
    END Sil_Tekrar_Eden_Kayitlar_Patient;

    PROCEDURE Sil_Tekrar_Eden_Kayitlar_Doctor IS
    BEGIN
        DELETE FROM Doctor
        WHERE ROWID NOT IN (
            SELECT MIN(ROWID)
            FROM Doctor
            GROUP BY Doctor_ID, Patient_ID, Department_ID, Age, Full_Name, Salary, Phone, Email, Specialization
```

```
Package Body created.
```

Tekrar eden veri oluşturduk



5	123-456-7890	Ahmet Yılmaz	M	A+	İstanbul, Türkiye	Mehmet Yılmaz	30
1	535-456-7890	Kasım Yılmaz	M	0+	İstanbul, Türkiye	Bülent Yılmaz	36
6	567-890-1234	Burak Aydın	M	A-	Antalya, Türkiye	Zeynep Aydın	28
101	123-456-7890	Ahmet Yılmaz	M	A+	İstanbul, Türkiye	Mehmet Yılmaz	30
102	123-456-7890	Ahmet Yılmaz	M	A+	İstanbul, Türkiye	Mehmet Yılmaz	30
103	234-567-8901	Ayşe Demir	F	B+	Ankara, Türkiye	Fatma Demir	25
104	234-567-8901	Ayşe Demir	F	B+	Ankara, Türkiye	Fatma Demir	25
105	345-678-9012	Mehmet Kaya	M	O-	İzmir, Türkiye	Ali Kaya	40
106	345-678-9012	Mehmet Kaya	M	O-	İzmir, Türkiye	Ali Kaya	40
2	545-456-7890	Muammer Yıl	M	AB+	Tokat, Türkiye	Ayşe Yıl	41
3	345-678-9012	Mehmet Kaya	M	O-	İzmir, Türkiye	Ali Kaya	40
4	456-789-0123	Elif Çelik	F	AB+	Bursa, Türkiye	Ayşe Çelik	35