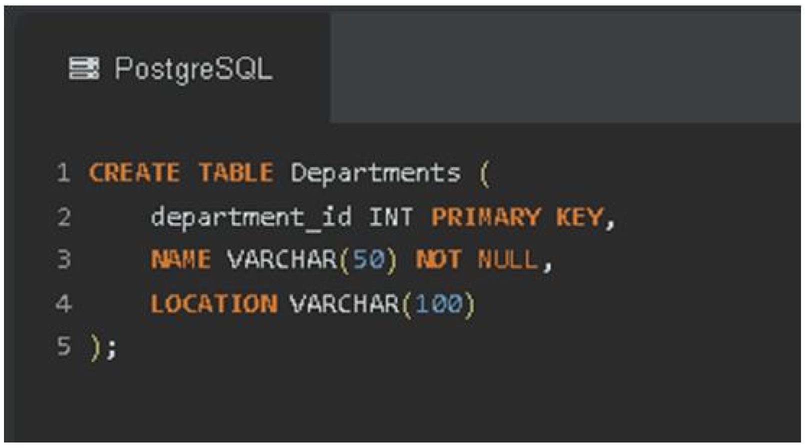
Advance SQL Project

Description: In this example, created 4 tables: Departments, Doctors, Patients, and Appointments. Each table has primary and foreign keys, as well as at least 2 not null and unique constraints. We have also created 2 procedures: CreateAppointment and GetAppointmentsByDoctor. The CreateAppointment procedure inserts a new appointment into the Appointments table, while the GetAppointmentsByDoctor procedure returns a list of appointments for a specific doctor. Finally, we have created 2 users with different levels of privileges: hospital_admin with all privileges and hospital_guest with only select privileges.

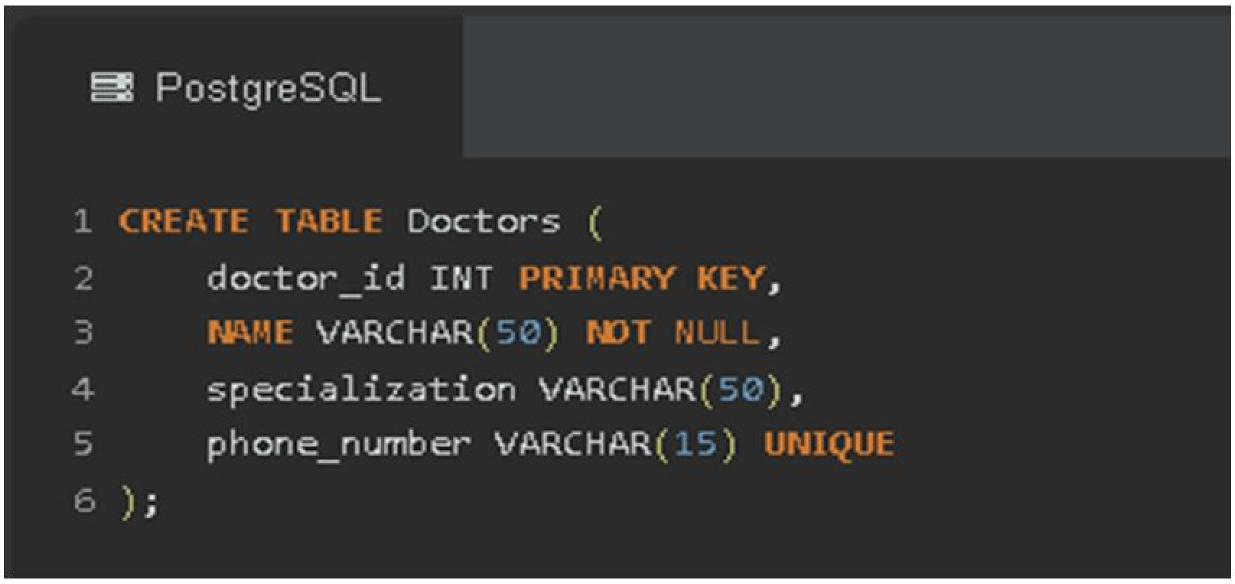
Solution:

Create Department Table



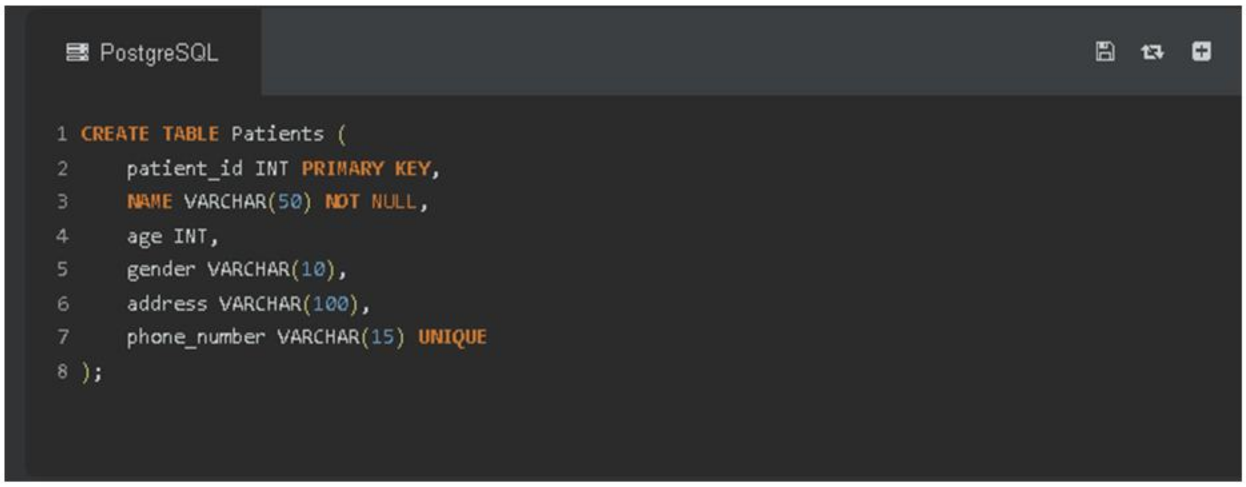
```
CREATE TABLE Departments (
    department_id INT PRIMARY KEY,
    NAME VARCHAR(50) NOT NULL,
    LOCATION VARCHAR(100)
);
```

Create Doctors Table



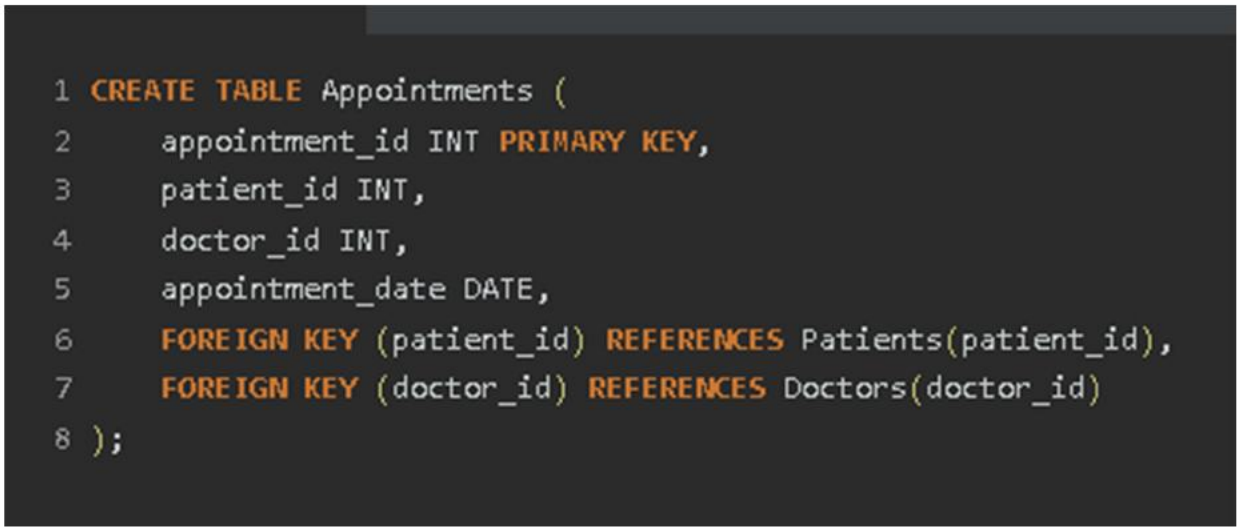
```
CREATE TABLE Doctors (
    doctor_id INT PRIMARY KEY,
    NAME VARCHAR(50) NOT NULL,
    specialization VARCHAR(50),
    phone_number VARCHAR(15) UNIQUE
);
```

Create Patients Table



```sql
CREATE TABLE Patients (
    patient_id INT PRIMARY KEY,
    NAME VARCHAR(50) NOT NULL,
    age INT,
    gender VARCHAR(10),
    address VARCHAR(100),
    phone_number VARCHAR(15) UNIQUE
);
```

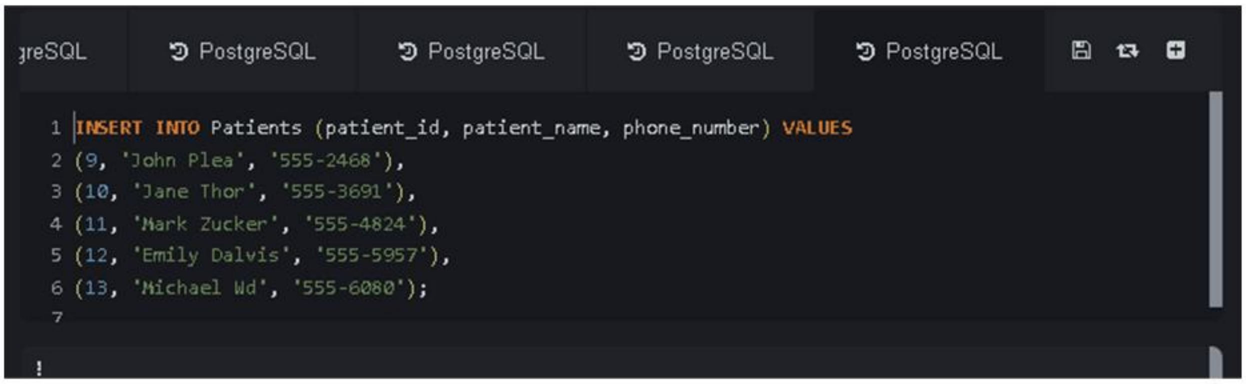
Create Appointments Table



```sql
CREATE TABLE Appointments (
    appointment_id INT PRIMARY KEY,
    patient_id INT,
    doctor_id INT,
    appointment_date DATE,
    FOREIGN KEY (patient_id) REFERENCES Patients(patient_id),
    FOREIGN KEY (doctor_id) REFERENCES Doctors(doctor_id)
);
```

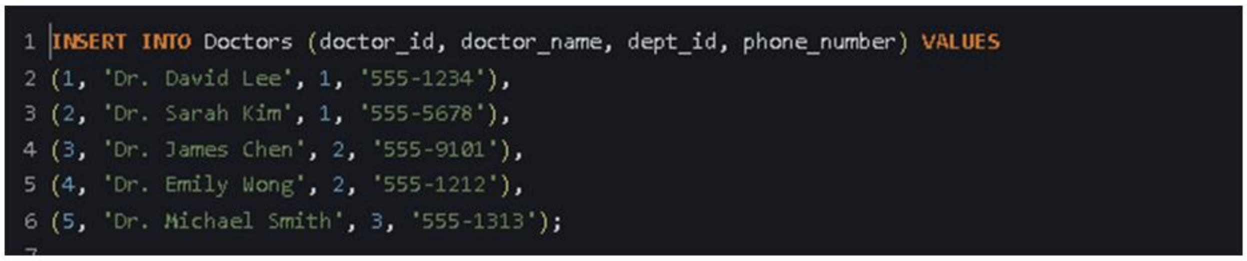
Insert Values:

A} Patients



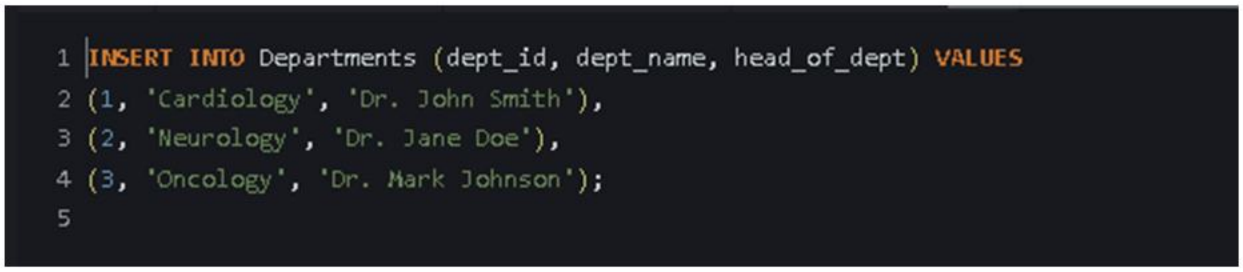
```
INSERT INTO Patients (patient_id, patient_name, phone_number) VALUES
(9, 'John Plea', '555-2468'),
(10, 'Jane Thor', '555-3691'),
(11, 'Mark Zucker', '555-4824'),
(12, 'Emily Dalvis', '555-5957'),
(13, 'Michael Wd', '555-6080');
```

B} Doctors



```
INSERT INTO Doctors (doctor_id, doctor_name, dept_id, phone_number) VALUES
(1, 'Dr. David Lee', 1, '555-1234'),
(2, 'Dr. Sarah Kim', 1, '555-5678'),
(3, 'Dr. James Chen', 2, '555-9101'),
(4, 'Dr. Emily Wong', 2, '555-1212'),
(5, 'Dr. Michael Smith', 3, '555-1313');
```

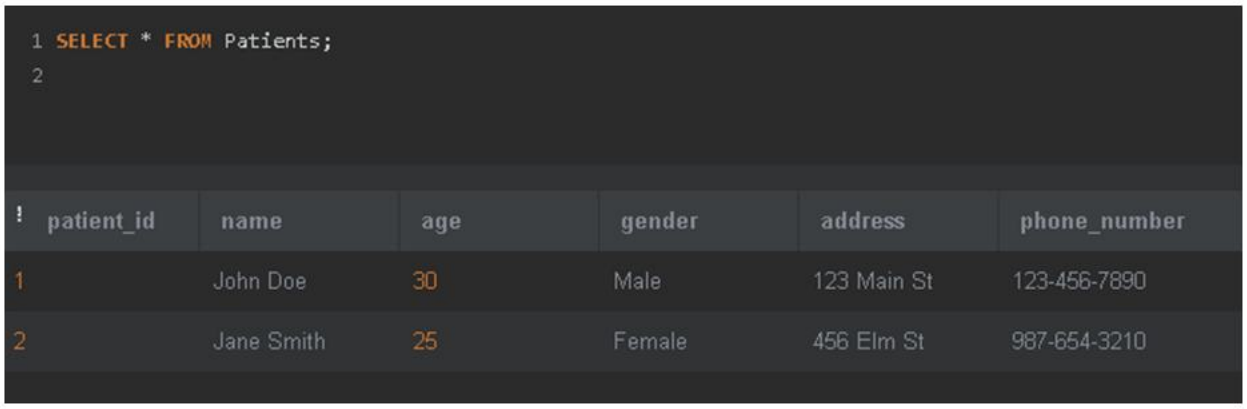
C} Departments



```
INSERT INTO Departments (dept_id, dept_name, head_of_dept) VALUES
(1, 'Cardiology', 'Dr. John Smith'),
(2, 'Neurology', 'Dr. Jane Doe'),
(3, 'Oncology', 'Dr. Mark Johnson');
```

Outputs for created Tables:

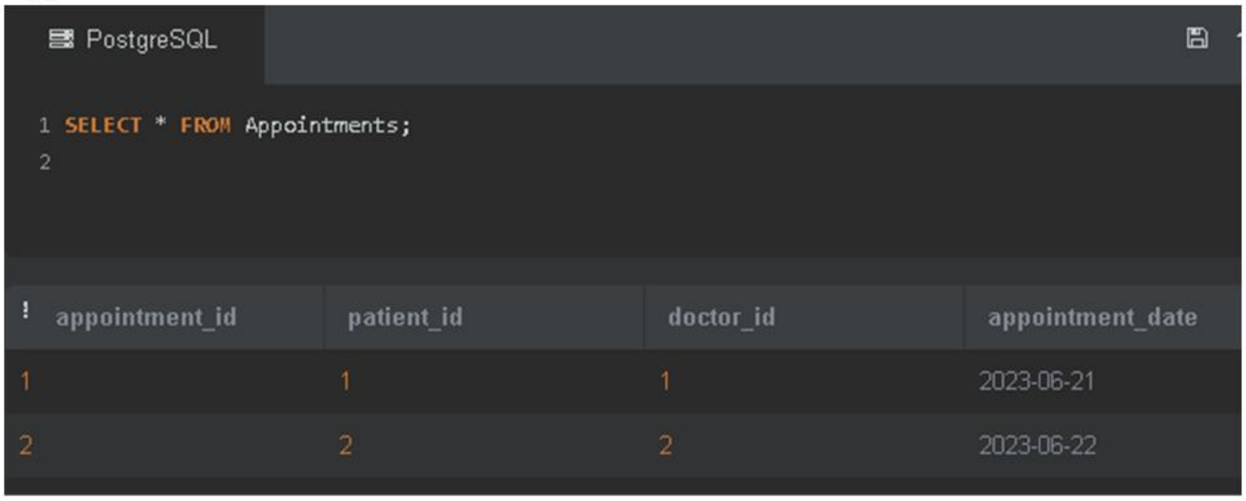
1. **Patients**



```
SELECT * FROM Patients;
```

```
patient_id   name         age   gender   address       phone_number
1            John Doe     30    Male     123 Main St   123-456-7890
2            Jane Smith   25    Female   456 Elm St    987-654-3210
```

2. **Appointments**

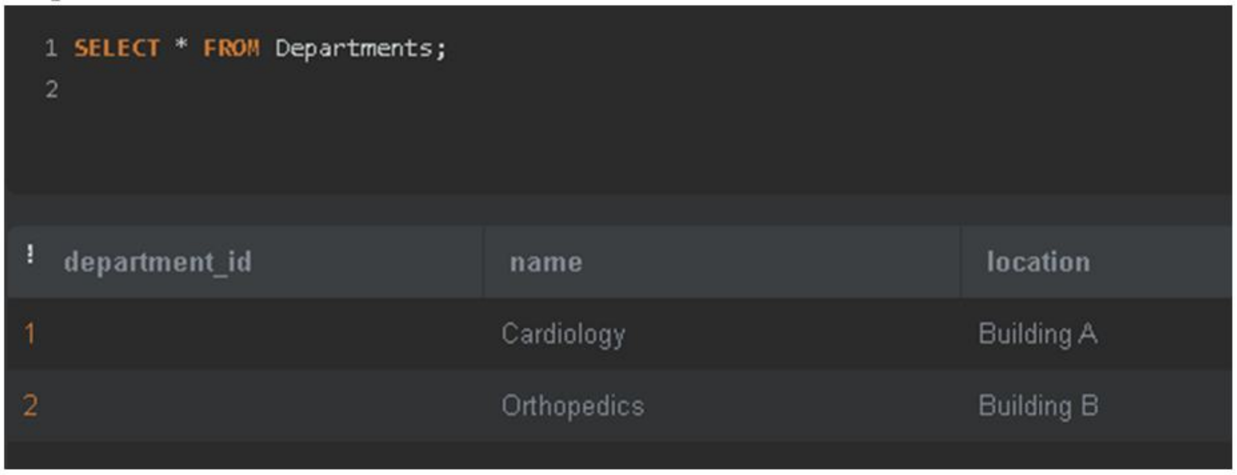


PostgreSQL

```
SELECT * FROM Appointments;
```

appointment_id	patient_id	doctor_id	appointment_date
1	1	1	2023-06-21
2	2	2	2023-06-22

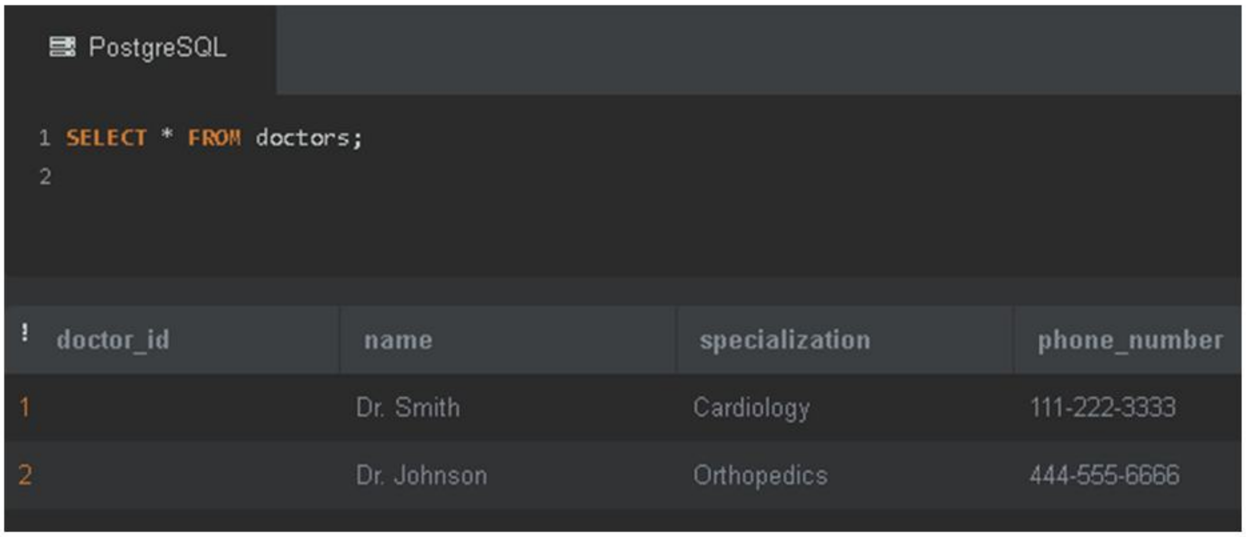
3. **Departments**



```
SELECT * FROM Departments;
```

department_id	name	location
1	Cardiology	Building A
2	Orthopedics	Building B

4. **Doctors**

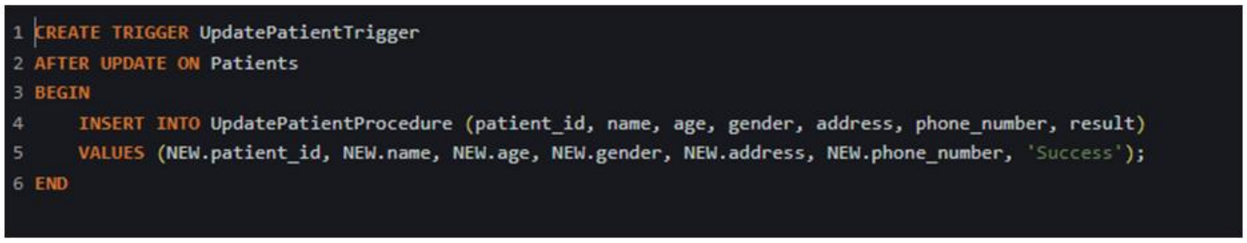


PostgreSQL

```
SELECT * FROM doctors;
```

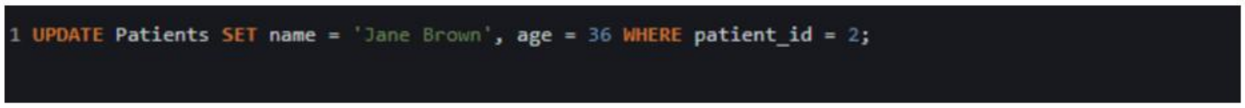
doctor_id	name	specialization	phone_number
1	Dr. Smith	Cardiology	111-222-3333
2	Dr. Johnson	Orthopedics	444-555-6666

Create a procedure to update a patient's information:

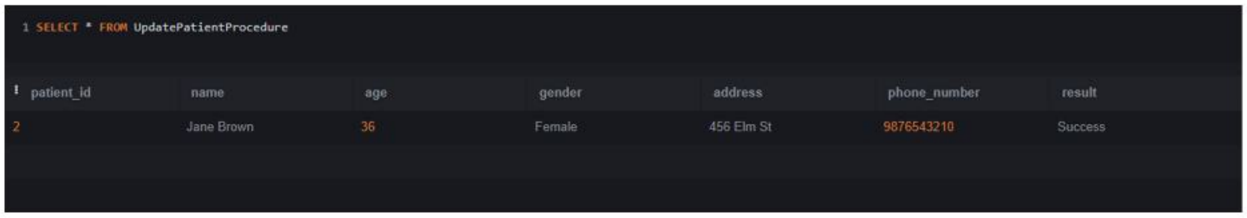


```
CREATE TRIGGER UpdatePatientTrigger
AFTER UPDATE ON Patients
BEGIN
    INSERT INTO UpdatePatientProcedure (patient_id, name, age, gender, address, phone_number, result)
    VALUES (NEW.patient_id, NEW.name, NEW.age, NEW.gender, NEW.address, NEW.phone_number, 'Success');
END
```

Output:



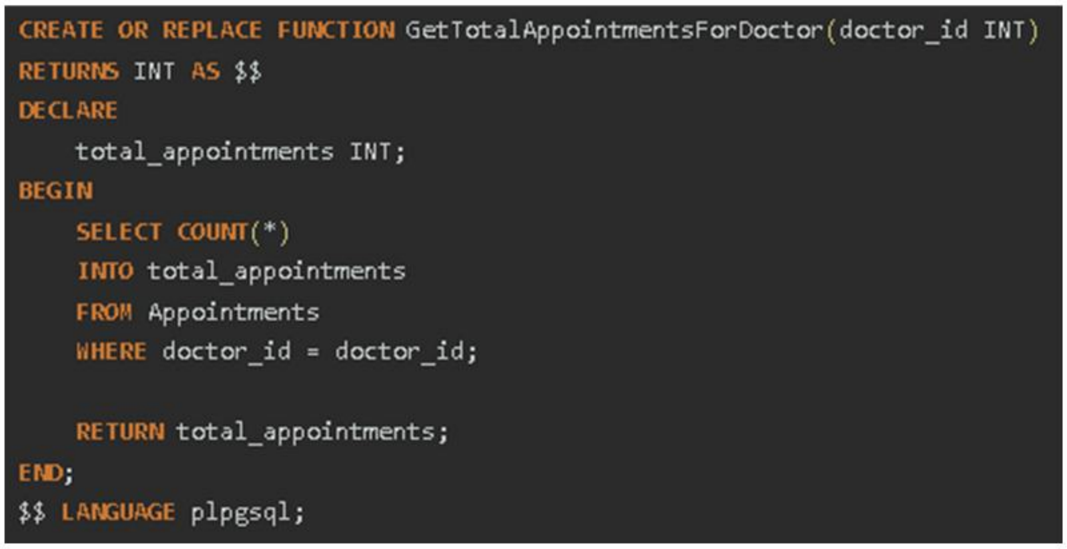
```
UPDATE Patients SET name = 'Jane Brown', age = 36 WHERE patient_id = 2;
```



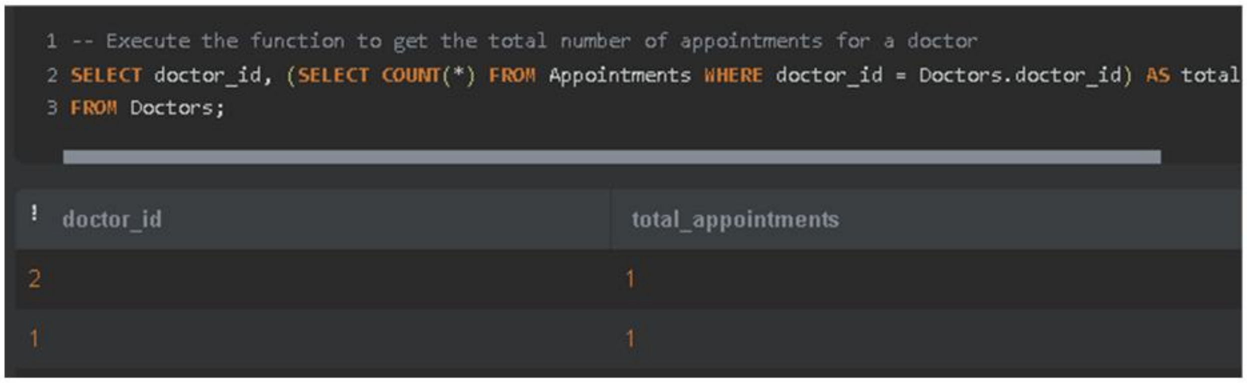
```
SELECT * FROM UpdatePatientProcedure
```

patient_id	name	age	gender	address	phone_number	result
2	Jane Brown	36	Female	456 Elm St	9876543210	Success

Create a function to get the total number of appointments for a doctor:



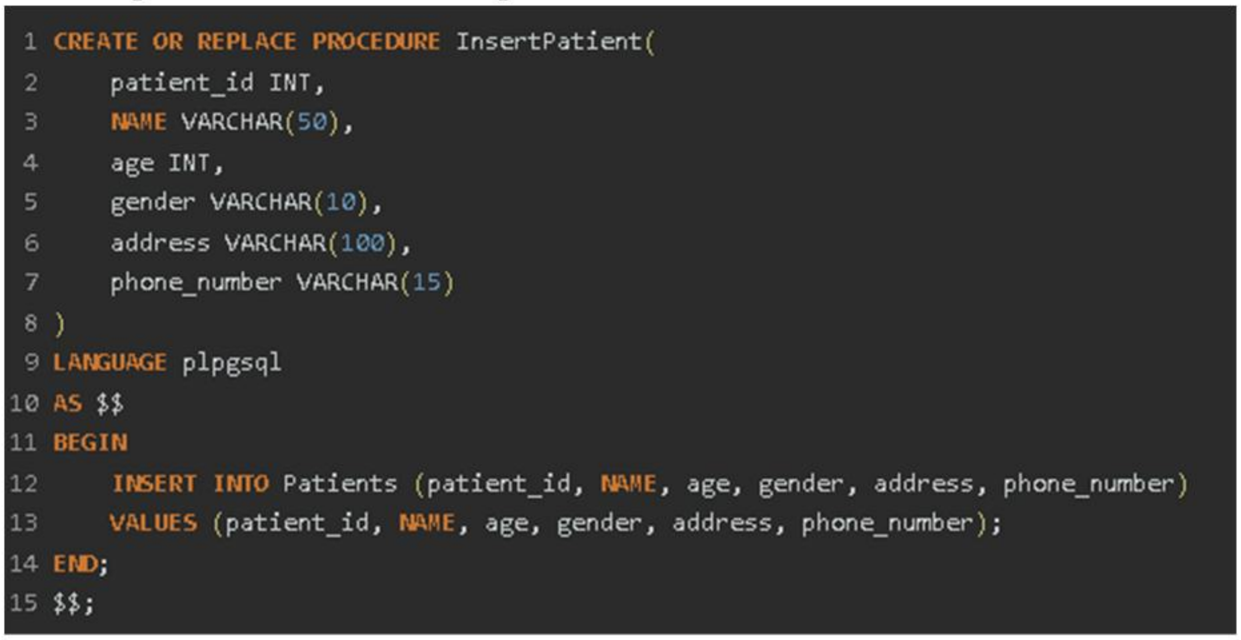
```
CREATE OR REPLACE FUNCTION GetTotalAppointmentsForDoctor(doctor_id INT)
RETURNS INT AS $$
DECLARE
    total_appointments INT;
BEGIN
    SELECT COUNT(*)
    INTO total_appointments
    FROM Appointments
    WHERE doctor_id = doctor_id;

    RETURN total_appointments;
END;
$$ LANGUAGE plpgsql;
```



```
-- Execute the function to get the total number of appointments for a doctor
SELECT doctor_id, (SELECT COUNT(*) FROM Appointments WHERE doctor_id = Doctors.doctor_id) AS total
FROM Doctors;
```

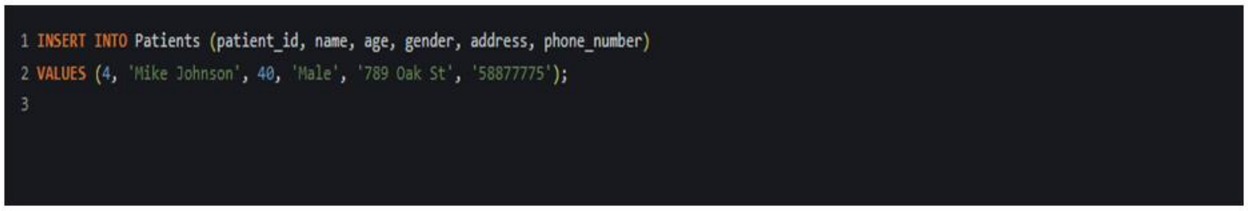
doctor_id	total_appointments
2	1
1	1

Create a procedure to insert a new patient

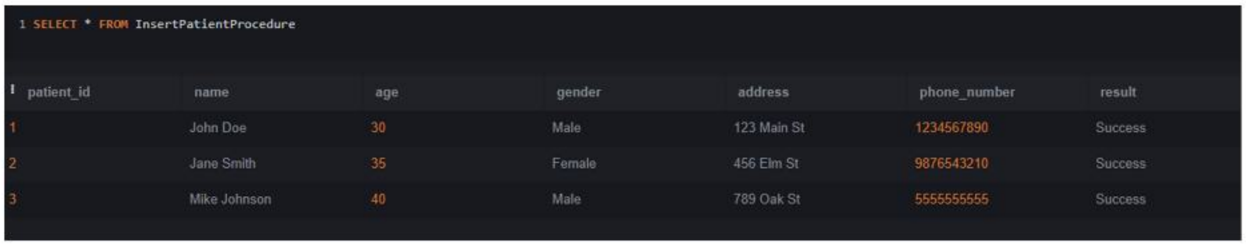


```
CREATE OR REPLACE PROCEDURE InsertPatient(
    patient_id INT,
    NAME VARCHAR(50),
    age INT,
    gender VARCHAR(10),
    address VARCHAR(100),
    phone_number VARCHAR(15)
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO Patients (patient_id, NAME, age, gender, address, phone_number)
    VALUES (patient_id, NAME, age, gender, address, phone_number);
END;
$$;
```

Output:



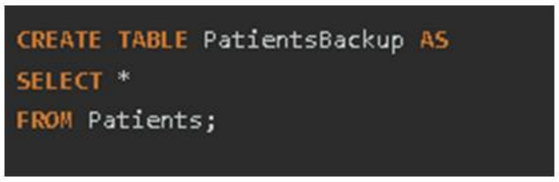
```
INSERT INTO Patients (patient_id, name, age, gender, address, phone_number)
VALUES (4, 'Mike Johnson', 40, 'Male', '789 Oak St', '58877775');
```



```
SELECT * FROM InsertPatientProcedure
```

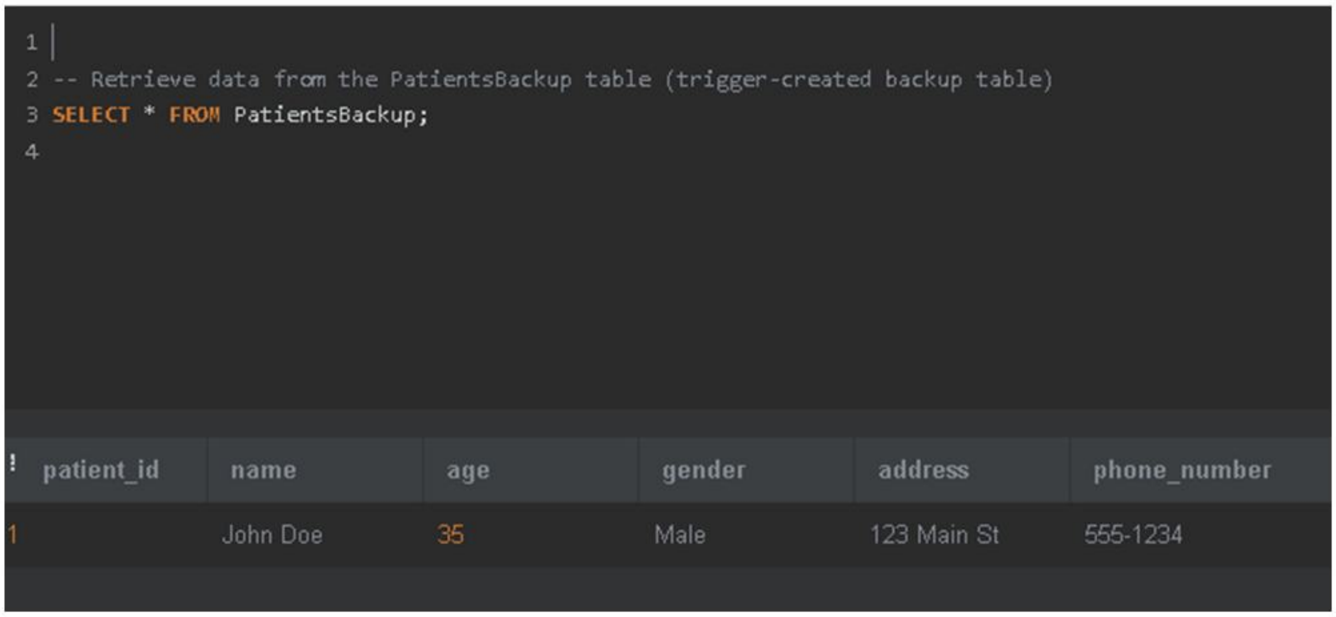
patient_id	name	age	gender	address	phone_number	result
1	John Doe	30	Male	123 Main St	1234567890	Success
2	Jane Smith	35	Female	456 Elm St	9876543210	Success
3	Mike Johnson	40	Male	789 Oak St	5555555555	Success

Create A Backup Table:



```sql
CREATE TABLE PatientsBackup AS
SELECT *
FROM Patients;
```

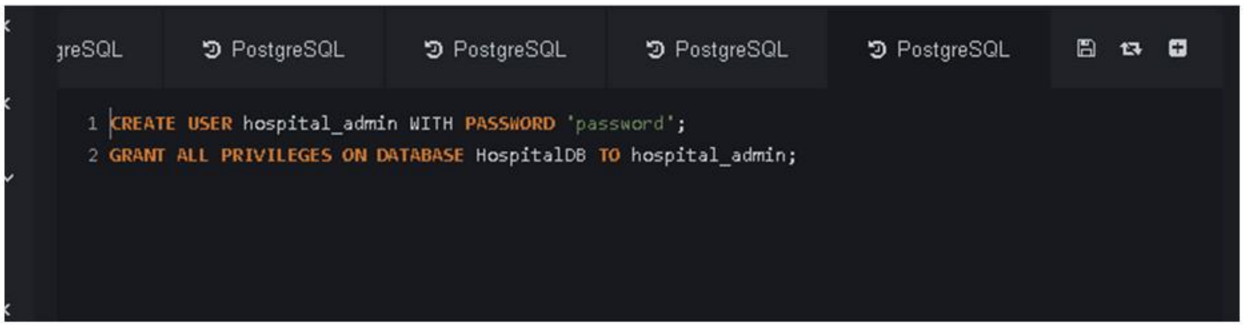
Output:



```sql

-- Retrieve data from the PatientsBackup table (trigger-created backup table)
SELECT * FROM PatientsBackup;

```

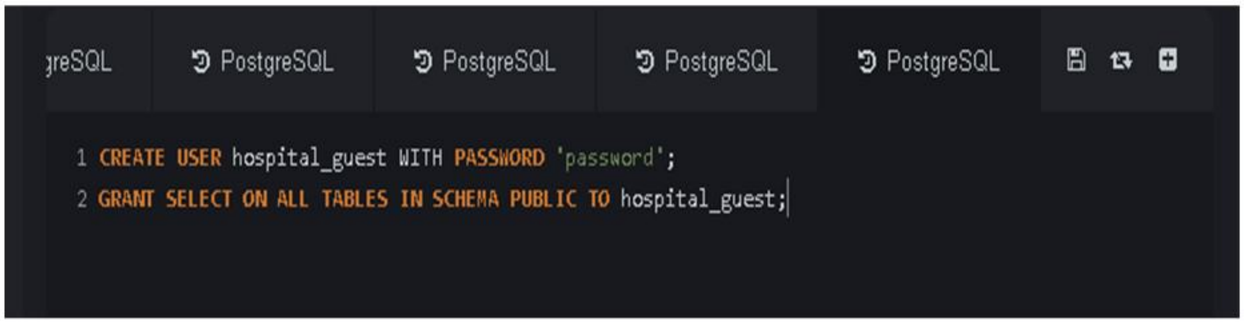
patient_id	name	age	gender	address	phone_number
1	John Doe	35	Male	123 Main St	555-1234

Create User for Admin:



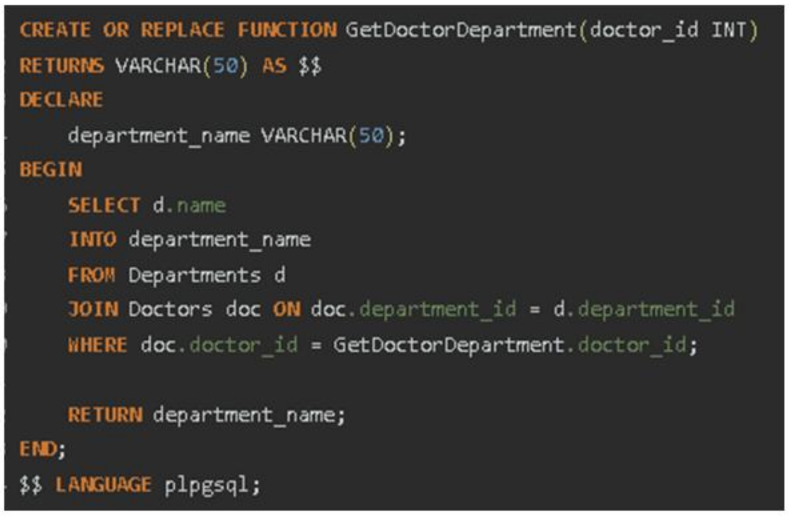
```sql
CREATE USER hospital_admin WITH PASSWORD 'password';
GRANT ALL PRIVILEGES ON DATABASE HospitalDB TO hospital_admin;
```

Create User for Guest:

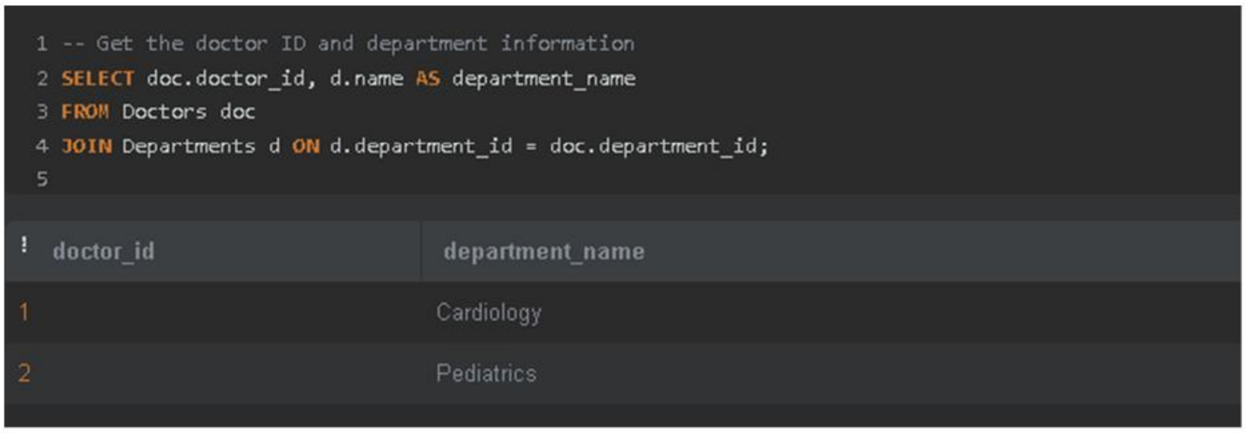


```sql
CREATE USER hospital_guest WITH PASSWORD 'password';
GRANT SELECT ON ALL TABLES IN SCHEMA PUBLIC TO hospital_guest;
```

Create a function to get the department of a doctor:



```
CREATE OR REPLACE FUNCTION GetDoctorDepartment(doctor_id INT)
RETURNS VARCHAR(50) AS $$
DECLARE
    department_name VARCHAR(50);
BEGIN
    SELECT d.name
    INTO department_name
    FROM Departments d
    JOIN Doctors doc ON doc.department_id = d.department_id
    WHERE doc.doctor_id = GetDoctorDepartment.doctor_id;

    RETURN department_name;
END;
$$ LANGUAGE plpgsql;
```



```
-- Get the doctor ID and department information
SELECT doc.doctor_id, d.name AS department_name
FROM Doctors doc
JOIN Departments d ON d.department_id = doc.department_id;

```

doctor_id	department_name
1	Cardiology
2	Pediatrics