

**CSE 414 DATABASE**

**SPRING 2024**

**HOMEWORK 2**

**MERT GÜRŞİMŞİR**

**1901042646**

# CONTENTS

[CONTENTS 2](#_Toc168226681)

[DESIGN 3](#_Toc168226682)

[SQL FUNCTIONS 5](#_Toc168226683)

[Table SQL Function 6](#_Toc168226684)

[For Loop Function 7](#_Toc168226685)

[Function with Variable 8](#_Toc168226686)

[TRIGGERS 9](#_Toc168226687)

[Trigger with “OLD” and “NEW” 9](#_Toc168226688)

[Trigger with “when” and “if” 10](#_Toc168226689)

[Trigger with “for each row” 12](#_Toc168226690)

[Drop a Trigger & Show Triggers 13](#_Toc168226691)

[TRANSACTIONS 15](#_Toc168226692)

[First Atomic Transcation 15](#_Toc168226693)

[Second Atomic Transaction 16](#_Toc168226694)

# DESIGN

In this homework, I am going to develop a management system database related to health sector step by step. For the building of database system for health sector operations, I have used PostgreSQL.

There are going to be 7 tables for my management system which are listed below:

* Patients

A close up of a data

Description automatically generated

* Doctors

A close up of a computer screen

Description automatically generated

* Appointments

A computer code with text

Description automatically generated

* Records

A computer code with text

Description automatically generated

* Departments

A close up of text

Description automatically generated

* DoctorDepartment

A computer screen shot of a computer code

Description automatically generated

Below the description of the relationships are listed:

|  |  |  |
| --- | --- | --- |
| **TABLES** | **CARDINALITY CONSTRAINT** | **DESCRIPTION** |
| Doctors – Appointments | One-to-Many | A doctor can have many appointments. |
| Doctors – Records | One-to-Many | A doctor can write many records. |
| Doctors – Departments | Many-to-Many | A doctor can work in many departments and a department can have many doctors. |
| Patients – Appointments | One-to-Many | A patient can have many appointments. |
| Patients – Records | One-to-Many | A patient can have many records. |

I am representing my schema diagram below:

A diagram of a program

Description automatically generated

# SQL FUNCTIONS

First of all, I am adding some values to the tables related to the functions that I am going to write:

A computer code with numbers and letters

Description automatically generated

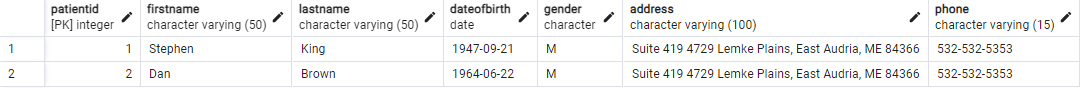
These tables look like these now:

Doctors

A screenshot of a phone

Description automatically generated

Patients



Appointments

A screenshot of a calendar

Description automatically generated

Now I am going to write functions needed for the management system one by one.

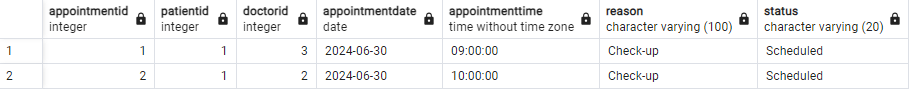
## Table SQL Function

I want to write a table SQL function that gets the ID of a patient and returns the appointments of that patient. I have written a function that is below and called it as you can see:

A screenshot of a computer screen

Description automatically generated

The result is this:



## For Loop Function

I have written an SQL for loop function that loops through all the appointments and writes down the information about each of the appointments. The function is like this:

A close-up of a computer screen

Description automatically generated

I have called it as follows:

A black and white text

Description automatically generated

As a result, I got the following:

A number and numbers on a white background

Description automatically generated

## Function with Variable

I have written a function that does the following: takes the patient ID as input, use a temporary variable that is total (means the total cost), calculates how many times a patient visits the hospital and multiply by 50 (each appointment costs 50 dollars). Then I have called the function at the bottom.

A computer screen shot of a program

Description automatically generated

The result is this:

A screenshot of a computer

Description automatically generated

# TRIGGERS

## Trigger with “OLD” and “NEW”

In PostgreSQL, there is no explicit definition such as “referencing old row as” and “referencing new row as”. The keywords OLD and NEW can be directly used to access the rows.

Therefore, I have used OLD and NEW in my trigger function.

For this trigger, I have created another table called AppointmentChanges. This table logs the appointments when their status have been changed and when they have been changed.

A white screen with black text

Description automatically generated

Trigger goes as follows:

A screenshot of a computer program

Description automatically generated



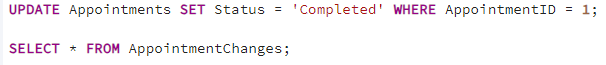
I am accessing the old row with OLD and new row with NEW.

Current Appointments table is as follows:

A screenshot of a calendar

Description automatically generated

Now assume the appointment with ID 1 has been cancelled. In that case this situation is going to be logged into the AppointmentChanges table. The query:



At the end, AppointmentChanges table looks like this:



*NOTE*: In PostgreSQL, triggers are written in the PL/pgSQL language, which requires the use of the ‘CREATE FUNCTION statement to define the trigger function. Therefore OLD and NEW keywords are used directly inside this trigger function.

## Trigger with “when” and “if”

In this trigger, I am aiming to prevent updating doctors’ specializations to a unvalid value. I have created a whitelist for the specializations and if a doctor’s specialization is going to be updated, this must be one of the specializations at the whitelist. Also the specialization must be different from the old specialization. With this way, trigger is triggered when the update is on specialization.

My trigger is as follows:

A screen shot of a computer

Description automatically generated

When I try to change the specialization of the doctor to a new value that is not in the whitelist, I expect from program to give an error.

Let’s update the doctor’s specialization, whose ID is 1, to ‘Anesthesiology’ which is not in the whitelist.



Result is as follows:

A black and white text

Description automatically generated

But if we try to change the specialization to a value that is in whitelist, update is going to be successful.



Then we can see specialization is changed:

A screenshot of a computer

Description automatically generated

## Trigger with “for each row”

Actually, with the former 2 triggers, we have used the “for each row”. Although in this trigger, the importance of it is going to be mentioned and understood deeply.

In this trigger, I am aiming to do this: Firstly I am going to add a column to the AppointmentCount. Without the trigger, when a new appointment occurs we have to increment the AppointmentCount of the doctor by hand. This is not an efficient approach.

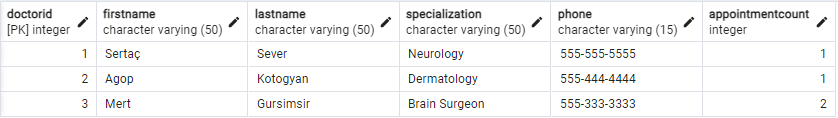


Now my aim is this: When now appointment is inserted into the Appointments table I am going to increment the related doctor’s AppointmentCount by 1 and in case of deletion of an appointment, I am going to decrease the number of AppointmentCount by 1. Also if the status of the appointment is updated, so it is not “Scheduled”, then we can decrease the AppointmentCount by 1 also.

A screenshot of a computer program

Description automatically generated

Let’s see how the Doctors and Appointments tables looks for now:



A screenshot of a phone

Description automatically generated

Let’s cancel the appointments of doctor with ID 3. In this case AppointmentCount of Mert should be 0 because there is no appointment for him:

A white background with red and black text

Description automatically generated

The result is as follows:

A screenshot of a computer

Description automatically generated

Thanks to “for each row” used in the trigger, it is executed for each row. So here 2 lines are updated at appointments and therefore trigger has executed 2 times.

## Drop a Trigger & Show Triggers

Firstly, let’s see the triggers. I want to see them with name, event object table, event manipulation (update, insert, delete, etc.), and action timing (before or after):



Result:

A screenshot of a computer

Description automatically generated

We can also see only the names of the triggers:



Result:

A screenshot of a computer

Description automatically generated

Now let’s drop some triggers. Firstly, drop the appointmentchange trigger and see the triggers again:

A close up of words

Description automatically generated

Result:

A screenshot of a computer

Description automatically generated

# TRANSACTIONS

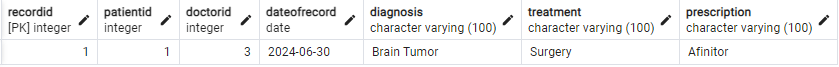
## First Atomic Transcation

With this transaction, I am aiming to inserting a new record to Records table and updating the status of an appointment to “Completed”. These both operations should be succeed together or fail together.

A close-up of a computer screen

Description automatically generated

Records table after transaction:



Appointments table after transaction:

A screenshot of a phone

Description automatically generated

## Second Atomic Transaction

In this transaction, I am aiming to adding a new department and assigning doctors to this department. These two operations should be succeed together or fail together.

A computer code with red and black text

Description automatically generated

Departments table after transaction:

A screenshot of a computer

Description automatically generated

DoctorDepartment table after transaction:

A screenshot of a computer

Description automatically generated