**Министерство науки и высшего образования Российской Федерации** федеральное государственное автономное образовательное учреждение высшего образования

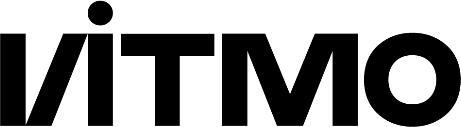
### «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

### Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL**»** по дисциплине **«Проектирование и реализация баз данных»**

Автор: Кадникова Е.М. Факультет: ИКТ Группа: К3241

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**Оглавление**

[Вариант 1. БД «Курсы» 3](#_bookmark0)

[Ход работы 4](#_bookmark1)

[Вывод 12](#_bookmark2)

**Цель работы:** овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД PostgreSQL, pgadmin 4.

### Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).

* 1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).
  2. Создать авторский триггер по варианту индивидуального задания.

**Вариант 7. БД «Курсы»**

Описание предметной области: Сеть учебных подразделений НОУ ДПО занимается организацией внебюджетного образования.

Имеется несколько образовательных программ краткосрочных курсов, предназначенных для определенных специальностей, связанных с программным обеспечением ИТ. Каждый программа имеет определенную длительность и свой перечень изучаемых дисциплин. Одна дисциплина может относиться к нескольким программам. На каждую программу может быть набрано несколько групп обучающихся.

По каждой дисциплине могут проводиться лекционные, лабораторные/практические занятия и практика определенном объеме часов. По каждой дисциплине и практике проводится аттестация в формате экзамен/дифзачет/зачет.

Необходимо хранить информацию по аттестации обучающихся.

Подразделение обеспечивает следующие ресурсы: учебные классы, лекционные аудитории и преподавателей. Необходимо составить расписание занятий.

БД должна содержать следующий минимальный набор сведений: Фамилия слушателя. Имя слушателя. Паспортные данные. Контакты. Код программы. Программа.  Тип программы. Объем часов. Номер группы. максимальное количество человек в группе (для набора). Дата начала обучения. Дата окончания обучения. Название дисциплины. Количество часов. Дата занятий. Номер пары. Номер аудитории. Тип аудитории. Адрес площадки. Вид занятий (лекционные, практические или лабораторные). Фамилия преподавателя. Имя и отчество преподавателя. Должность преподавателя. Дисциплины, которые может вести преподаватель.

**Задание 1.1 (ЛР 1 БД).**  Выполните инфологическое моделирование базы данных системы. (Ограничения задать самостоятельно.)

**Задание 1.2.** Создайте логическую модель БД, используя ИЛМ (задание 1.1).  Используйте необходимые средства поддержки целостности данных в СУБД.

**Задание 2.** Создать запросы:

* Вывести все номера групп и программы, где количество слушателей меньше 10.
* Вывести список преподавателей с указанием количества программ, где они преподавали за истекший год.
* Вывести список преподавателей, которые не проводят занятия на третьей паре ни в один из дней недели.
* Вывести список свободных лекционных аудиторий на ближайший понедельник.
* Вычислить общее количество обучающихся по каждой программе за последний год.
* Вычислить среднюю загруженность компьютерных классов в неделю за последний месяц (в часах).
* Найти самые популярные программы за последние 3 года.

**Задание 3.** Создать представление:

* для потенциальных слушателей, содержащее перечень специальностей, изучаемых на них дисциплин и количество часов;
* общих доход по каждой программе за последний год.

**Задание 4.** Создать хранимые процедуры:

* Для получения расписания занятий для групп на определенный день недели.
* Записи на курс слушателя.
* Получения перечня свободных лекционных аудиторий на любой день недели. Если свободных аудиторий не имеется, то выдать соответствующее сообщение.

**Задание 5.** Создать необходимые триггеры.

# Ход работы:

1. Процедура для получения расписания занятий для групп на определенный день недели:

CREATE OR REPLACE FUNCTION lab3.get\_schedule\_for\_weekday(weekday\_param VARCHAR(32))

RETURNS TABLE (lesson\_time VARCHAR(5), lesson\_type VARCHAR(24), subject VARCHAR(48), teacher TEXT, adress VARCHAR(36), auditorium INTEGER)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY (

SELECT l.lesson\_time, s.lesson\_type, sub.name\_subject, concat(t.surname\_teacher,' ',t.name\_teacher,' ', t.middle\_name\_teacher) AS teacher, a.adress, a.number\_aud

FROM lab3."Lesson" l

JOIN lab3."Schedule" s

ON l.id\_lesson = s.id\_lesson

JOIN lab3."Teacher" t

ON s.id\_teacher = t.id\_teacher

JOIN lab3."Study subject on programm" st

ON s.id\_subject\_on\_programm = st.id\_subject\_on\_programm

JOIN lab3."Study subject" sub

ON st.id\_subject = sub.id\_subject

JOIN lab3."Auditorium" a

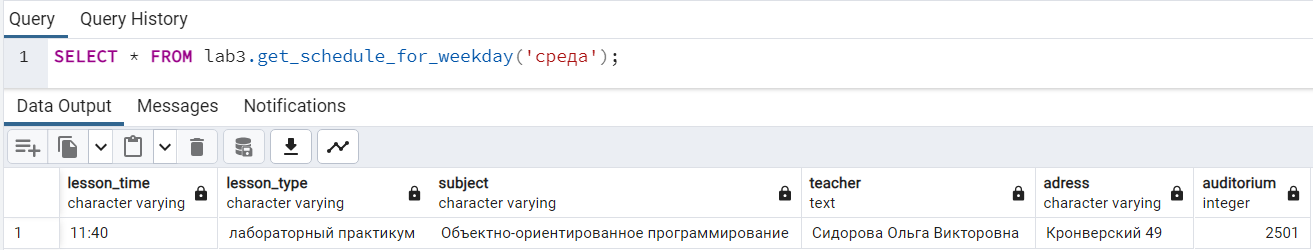
ON s.id\_aud = a.id\_aud

WHERE l.weekday = weekday\_param);

END;

$$





1. Процедура для записи слушателя на курс:

CREATE OR REPLACE PROCEDURE lab3.join\_course(id\_param INTEGER, surname\_param VARCHAR(24), name\_param VARCHAR(24), middle\_name\_param VARCHAR(24), passport\_param INTEGER, contacts VARCHAR(24), id\_programm\_param INTEGER, type\_recruit\_param VARCHAR(24))

LANGUAGE plpgsql

AS $$

BEGIN

IF NOT EXISTS (SELECT \* FROM lab3."Student" s WHERE s.id\_student = id\_param) THEN

INSERT INTO lab3."Student" VALUES (id\_param, surname\_param, name\_param, middle\_name\_param, passport\_param, contacts);

END IF;

IF NOT EXISTS (SELECT \* FROM lab3."Student in group" sg JOIN lab3."Group on programm" g ON sg.id\_group = g.id\_group WHERE sg.id\_student = id\_param AND g.id\_programm = id\_programm\_param) THEN

INSERT INTO lab3."Student in group"

VALUES

(id\_param,

(SELECT g.id\_group FROM lab3."Group on programm" g JOIN lab3."Student in group" sg ON g.id\_group = sg.id\_group WHERE (SELECT COUNT (\*) FROM lab3."Student in group" sg WHERE g.id\_group = sg.id\_group) < g.max\_people\_count ORDER BY g.id\_group LIMIT 1),

(SELECT education\_document + 1 FROM lab3."Student in group" ORDER BY education\_document DESC LIMIT 1),

(SELECT id\_student\_in\_group + 1 FROM lab3."Student in group" ORDER BY id\_student\_in\_group DESC LIMIT 1),

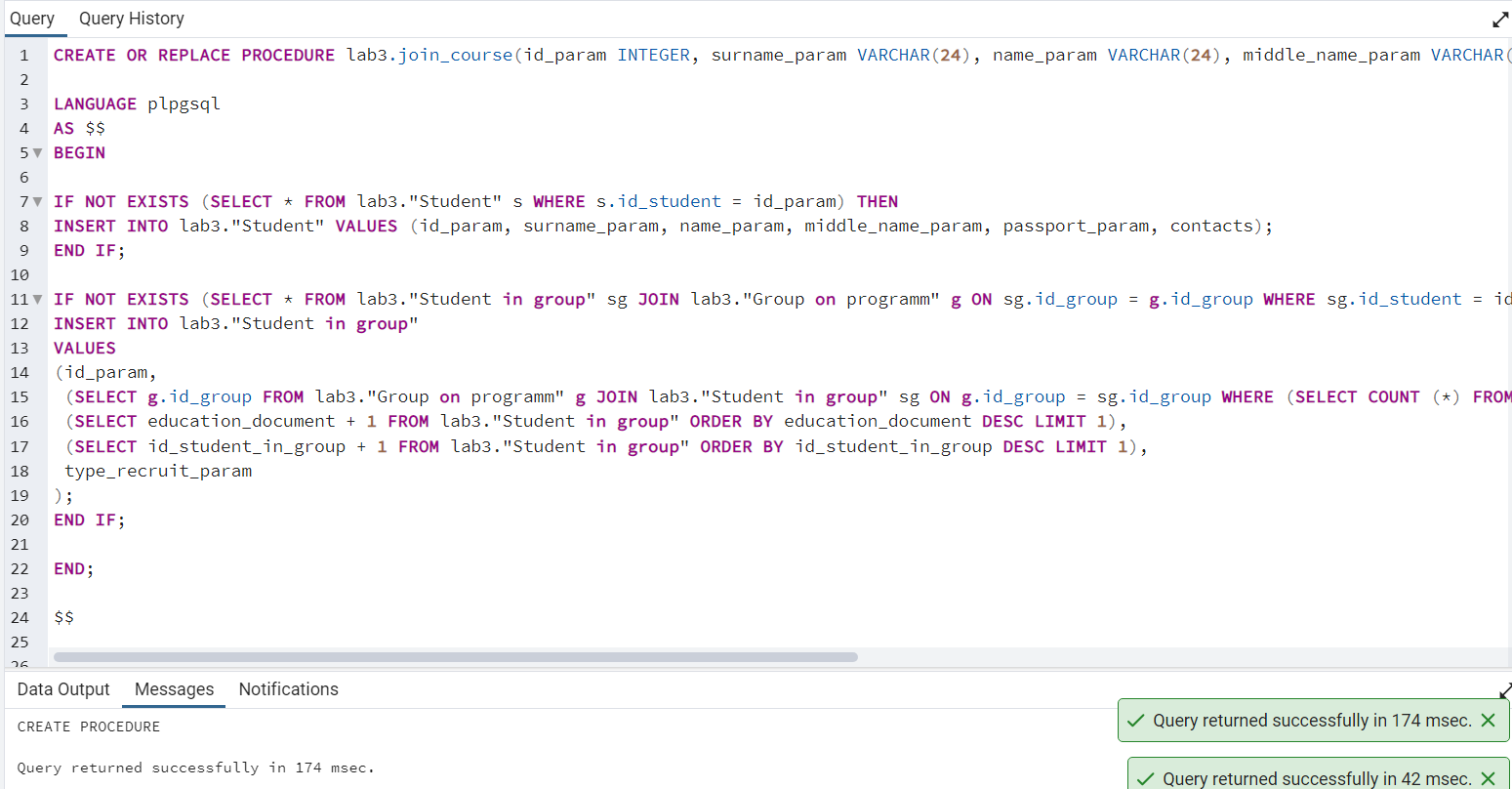
type\_recruit\_param

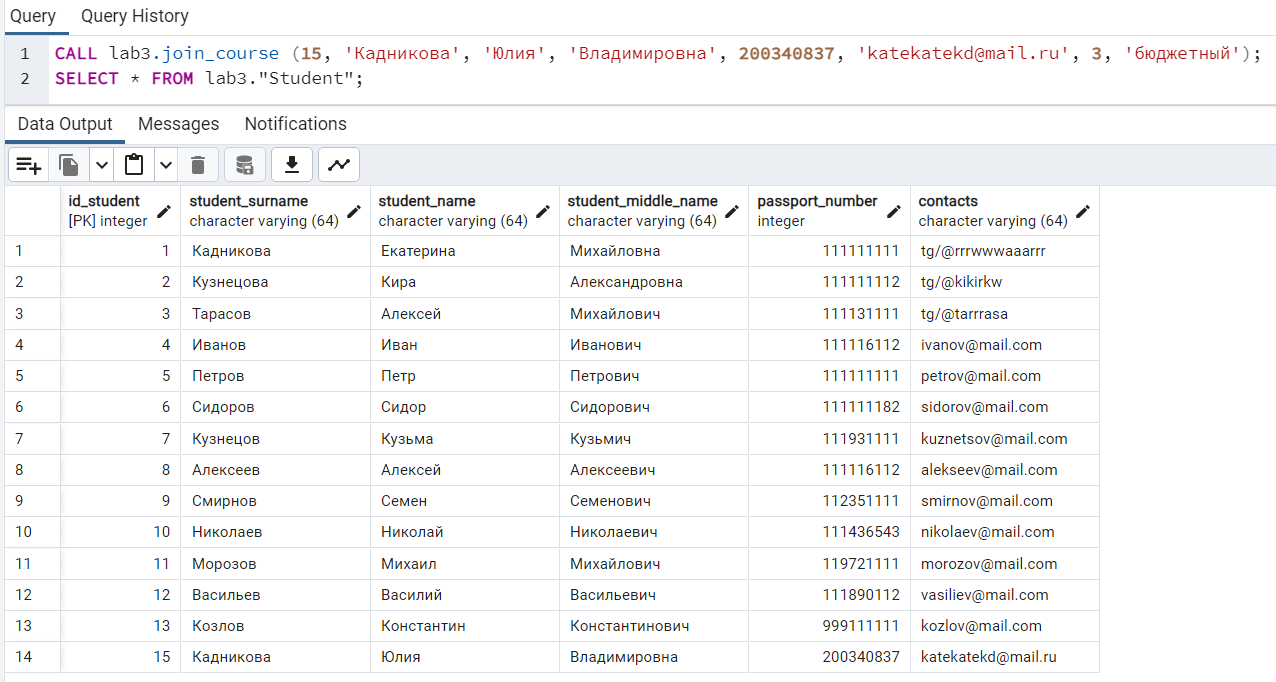
);

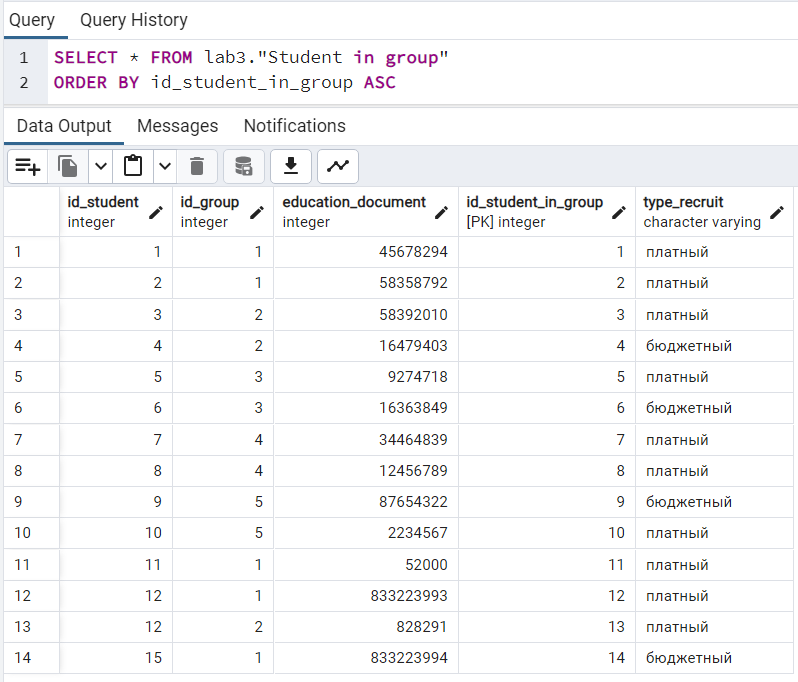
END IF;

END;

$$







1. Процедура для получения перечня свободных лекционных аудиторий на любой день недели:

CREATE OR REPLACE FUNCTION lab3.get\_free\_auds\_for\_weekday(weekday\_param VARCHAR(16))

RETURNS TABLE(adress VARCHAR(48), aud\_type VARCHAR(24), number INTEGER)

LANGUAGE plpgsql

AS $$

BEGIN

RETURN QUERY (SELECT a.adress, a.type\_aud, a.number\_aud

FROM lab3."Auditorium" a

WHERE id\_aud NOT IN (SELECT DISTINCT a.id\_aud FROM lab3."Auditorium" a

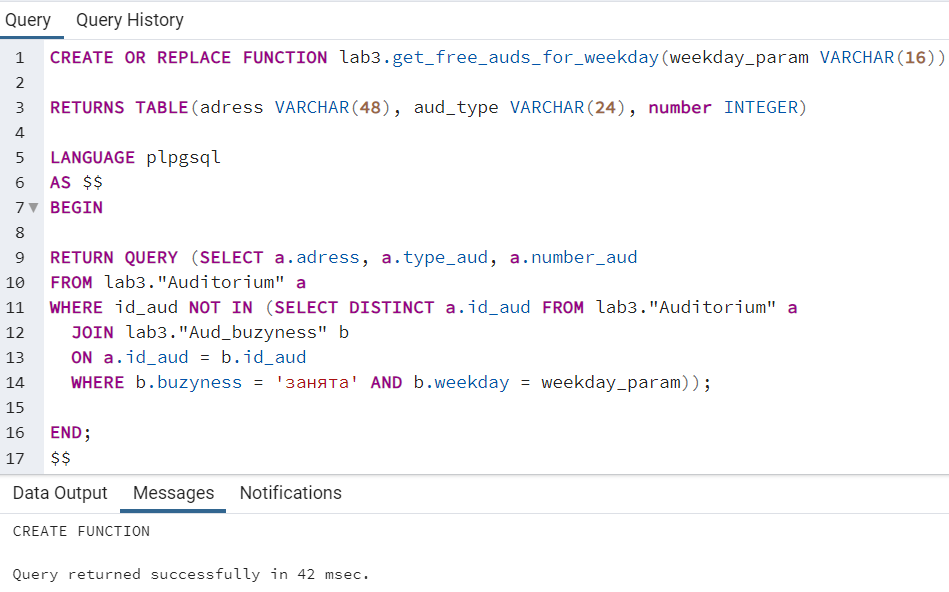
JOIN lab3."Aud\_buzyness" b

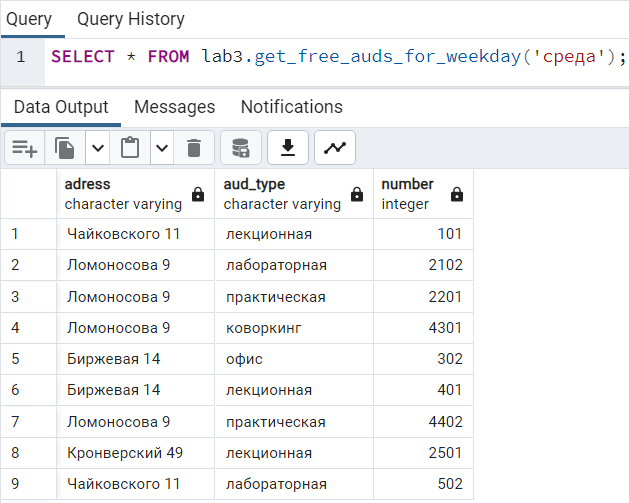
ON a.id\_aud = b.id\_aud

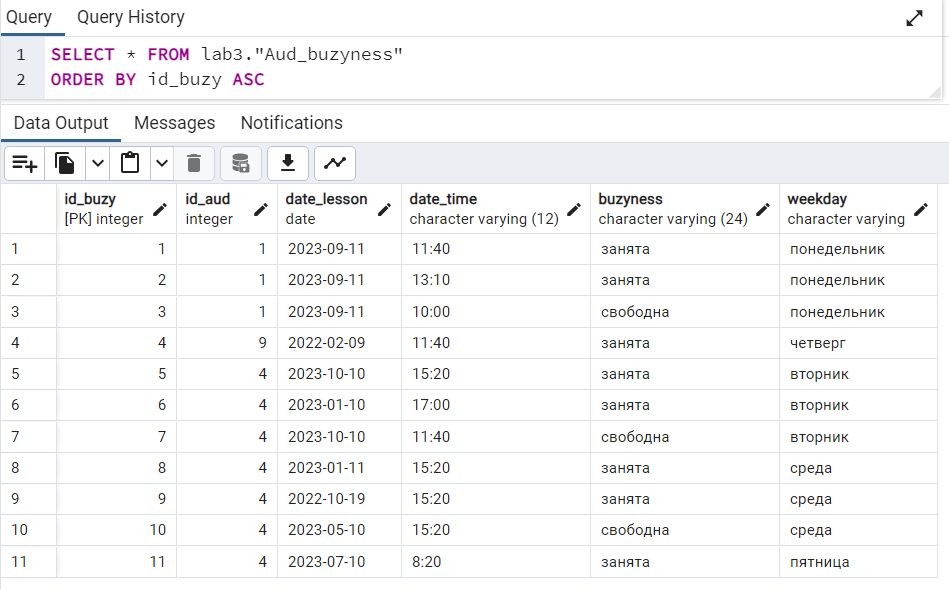
WHERE b.buzyness = 'занята' AND b.weekday = weekday\_param));

END;

$$







1. Модифицировать триггер на проверку корректности входа и выхода сотрудника

(имеющиеся проблемы: может быть отрицательное время работы, человек зашел/вышел в будущем)

create or replace function fn\_check\_time\_punch() returns trigger as $psql$ begin

if

new.is\_out\_punch = (select tps.is\_out\_punch from time\_punch tps

where tps.employee\_id = new.employee\_id order by tps.id desc limit 1 )

or

new.punch\_time>now()

or

new.punch\_time <= (select tps.punch\_time from time\_punch tps

where tps.employee\_id = new.employee\_id order by tps.id desc limit 1 )

then return null;

end if; return new;

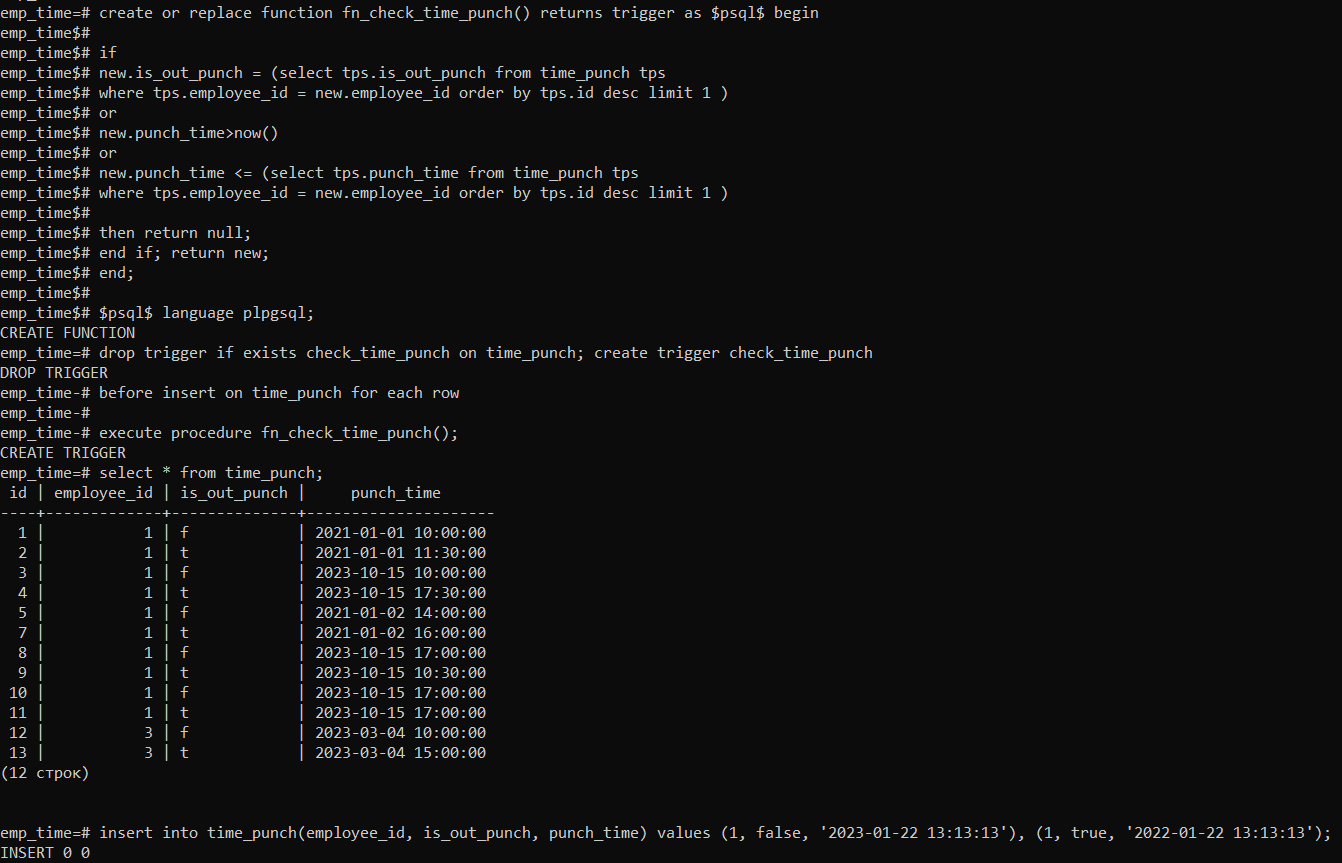
end;

$psql$ language plpgsql;

drop trigger if exists check\_time\_punch on time\_punch; create trigger check\_time\_punch

before insert on time\_punch for each row

execute procedure fn\_check\_time\_punch();



1. Авторский триггер для обновления списка студентов при отчислении с курса:

create or replace function lab3.drop\_student\_from\_course()

returns trigger

language plpgsql

as $$

begin

delete from lab3."Student" s where s.id\_student not in (select sg.id\_student from lab3."Student in group" sg);

return new;

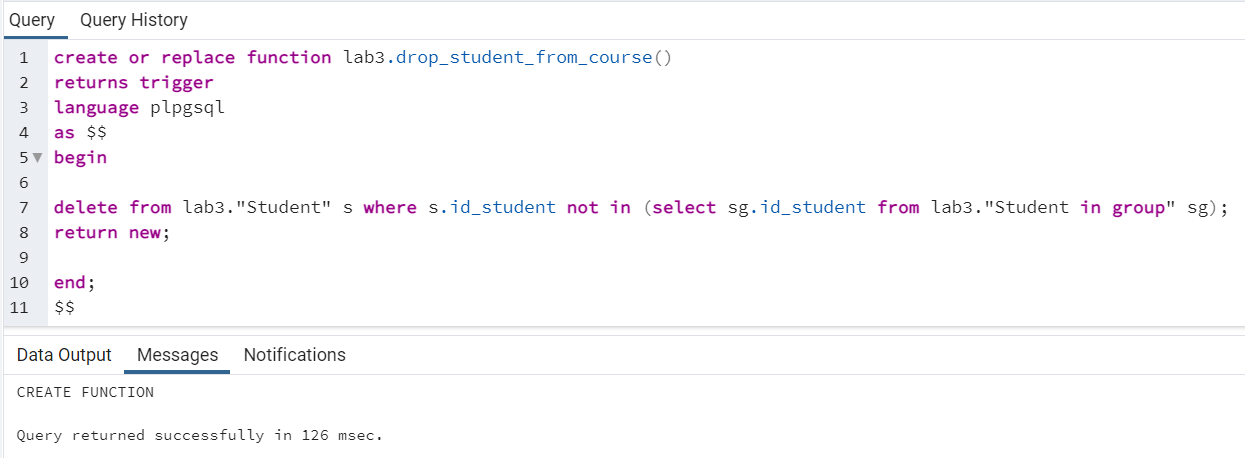
end;

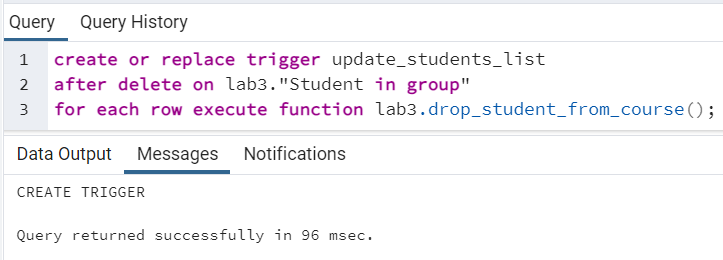
$$

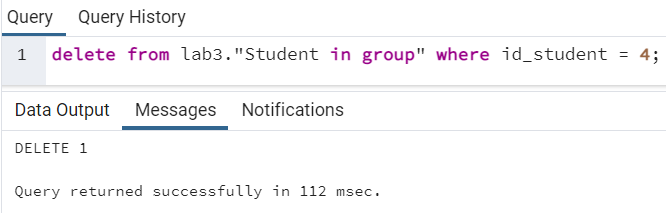
create or replace trigger update\_students\_list

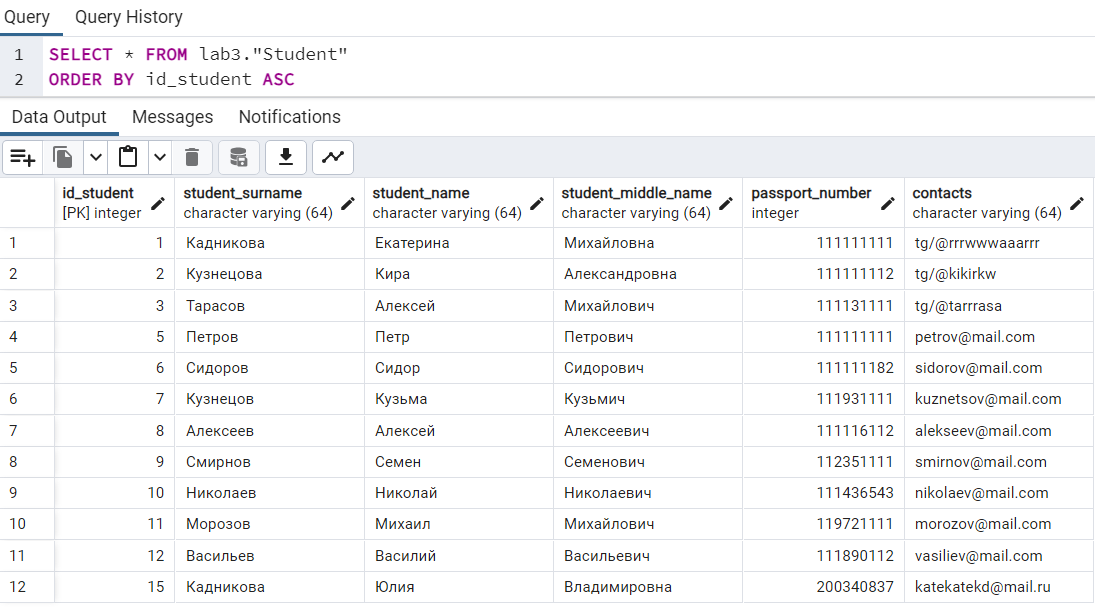
after delete on lab3."Student in group"

for each row execute function lab3.drop\_student\_from\_course();









# Вывод

## В ходе выполнения данной лабораторной работы я научилась создавать и работать с процедурами, функциями и триггерами в PostgreSQL.