Бюджетное учреждение высшего образования Ханты-Мансийского автономного округа – Югры

«СУРГУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА АВТОМАТИЗИРОВАННЫХ СИСТЕМ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ

ОТЧЕТ

ЛАБОРАТОРНАЯ РАБОТА №8

ПО ДИСЦИПЛИНЕ: «СОВРЕМЕННЫЕ ПРОМЫШЛЕННЫЕ СУБД»

Выполнил: студент группы №606-11

Ахметова Э. Р.

Проверил: ст. преподаватель

Гавриленко А. В.

Сургут 2025

1. Определите количество системных хранимых процедур.

SELECT

pr.proname AS procedure\_name,

(SELECT COUNT(\*)

FROM pg\_proc pr2

JOIN pg\_namespace ns2 ON pr2.pronamespace = ns2.oid

WHERE ns2.nspname = 'pg\_catalog' AND pr2.proname ~ '^pg\_.\*') AS proc\_sum

FROM pg\_proc pr

JOIN pg\_namespace ns ON pr.pronamespace = ns.oid

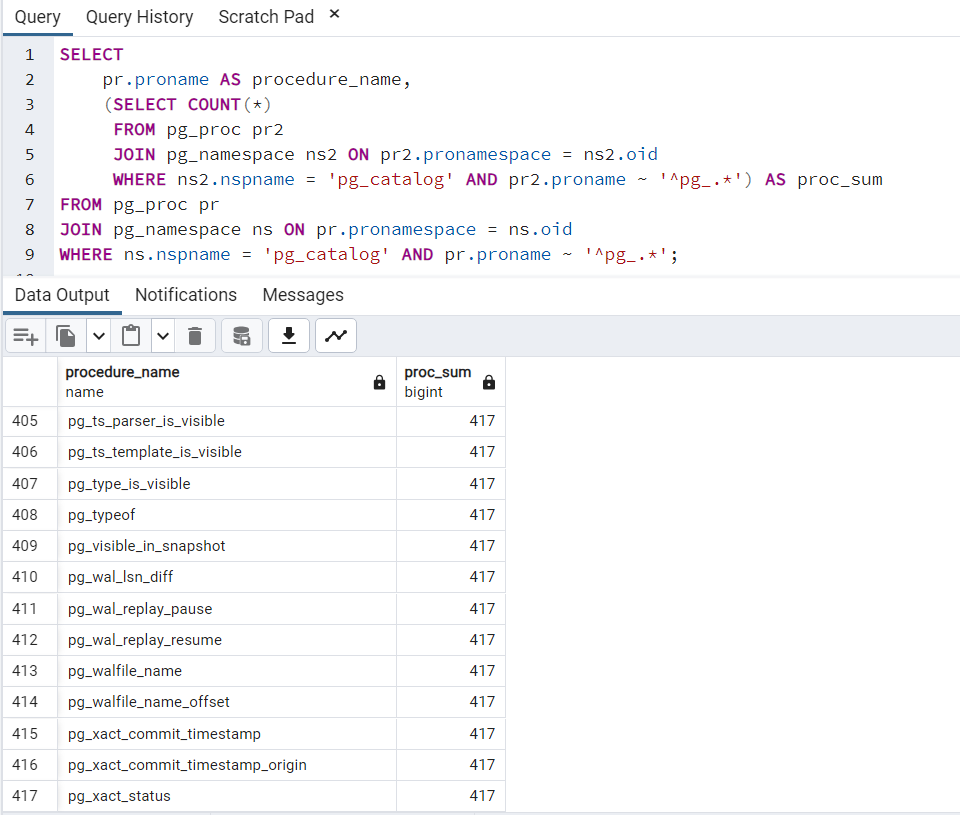
WHERE ns.nspname = 'pg\_catalog' AND pr.proname ~ '^pg\_.\*';  
Способ проще:

SELECT

pr.proname AS procedure\_name

FROM pg\_proc pr

WHERE pr.proname ~ '^pg\_.\*';



1. Создайте хранимую процедуру определяющую время года для заданной даты: осень, зима, весна или лето.

CREATE OR REPLACE PROCEDURE procedure1(

IN input\_date DATE,

OUT season TEXT

)

LANGUAGE plpgsql

AS $$

BEGIN

CASE EXTRACT(MONTH FROM input\_date)

WHEN 12, 1, 2 THEN

season := 'Зима';

WHEN 3, 4, 5 THEN

season := 'Весна';

WHEN 6, 7, 8 THEN

season := 'Лето';

WHEN 9, 10, 11 THEN

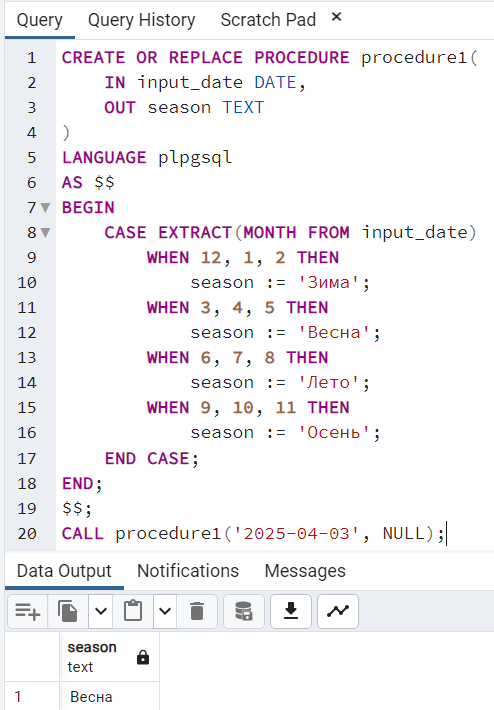
season := 'Осень';

END CASE;

END;

$$;

CALL procedure1('2025-04-03', NULL);



1. Создайте хранимую процедуру определяющую по дате рождения пользователя, исполнилось ему или нет полных 16 лет.

CREATE OR REPLACE PROCEDURE procedure2(

IN input\_date DATE,

OUT age BOOLEAN,

OUT age\_message TEXT

)

LANGUAGE plpgsql

AS $$

BEGIN

IF input\_date <= (CURRENT\_DATE - INTERVAL '16 years') THEN

age := TRUE;

age\_message := 'Пользователю исполнилось 16 лет';

ELSE

age := FALSE;

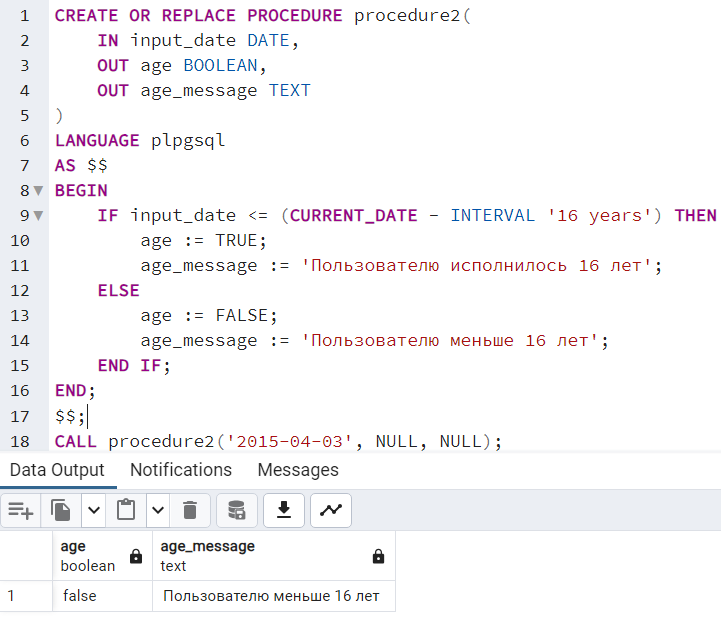
age\_message := 'Пользователю меньше 16 лет';

END IF;

END;

$$;

CALL procedure2('2015-04-03', NULL, NULL);



1. Создайте хранимую процедуру определяющую, является ли строка символов адресом электронной почты. Сделайте так, чтобы исходный текст процедуры нельзя было просмотреть.

CREATE OR REPLACE PROCEDURE procedure3(

IN input\_email TEXT

)

LANGUAGE plpgsql

SECURITY DEFINER

AS $$

DECLARE

is\_valid BOOLEAN;

BEGIN

is\_valid := input\_email ~ '^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9-]+(?:\.[a-zA-Z0-9-]+)+$';

IF is\_valid THEN

RAISE NOTICE 'Строка является эл. почтой';

ELSE

RAISE NOTICE 'Строка не является эл. почтой';

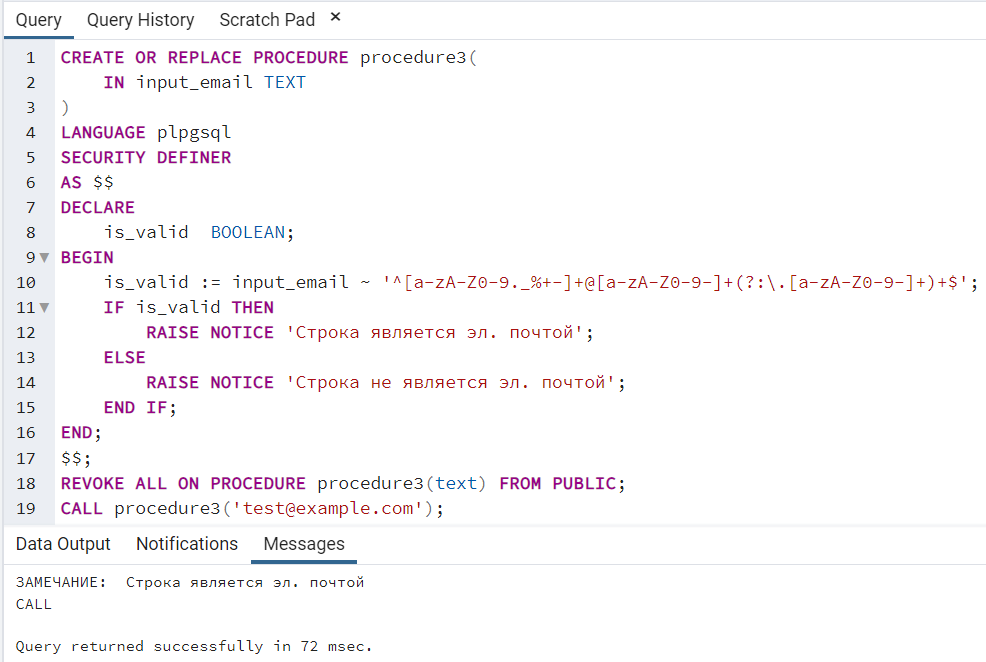
END IF;

END;

$$;

REVOKE ALL ON PROCEDURE procedure3(text) FROM PUBLIC;

CALL procedure3('test@example.com');



1. Создайте хранимую процедуру добавляющую к вашей базе данных новую таблицу Пользователи если она не существует. В таблице должны быть атрибуты Id, Login, Password, Email и вторичный ключ указывающий на одну из ваших таблиц где фигурирует понятие человек (назовем ее условно Клиенты).

CREATE OR REPLACE PROCEDURE procedure4()

LANGUAGE plpgsql

AS $$

BEGIN

IF NOT EXISTS (

SELECT 1

FROM information\_schema.tables

WHERE table\_schema = 'public'

AND table\_name = 'Пользователи'

) THEN

CREATE TABLE public."Пользователи" (

"Id" SERIAL PRIMARY KEY,

"Login" VARCHAR(50) NOT NULL UNIQUE,

"Password" VARCHAR(100) NOT NULL,

"Email" VARCHAR(100) NOT NULL UNIQUE,

"ID\_Клиента" INTEGER,

CONSTRAINT fk\_employee

FOREIGN KEY ("ID\_Клиента")

REFERENCES public."Клиенты"("ID")

ON DELETE SET NULL

);

RAISE NOTICE 'Таблица "Пользователи" создана';

ELSE

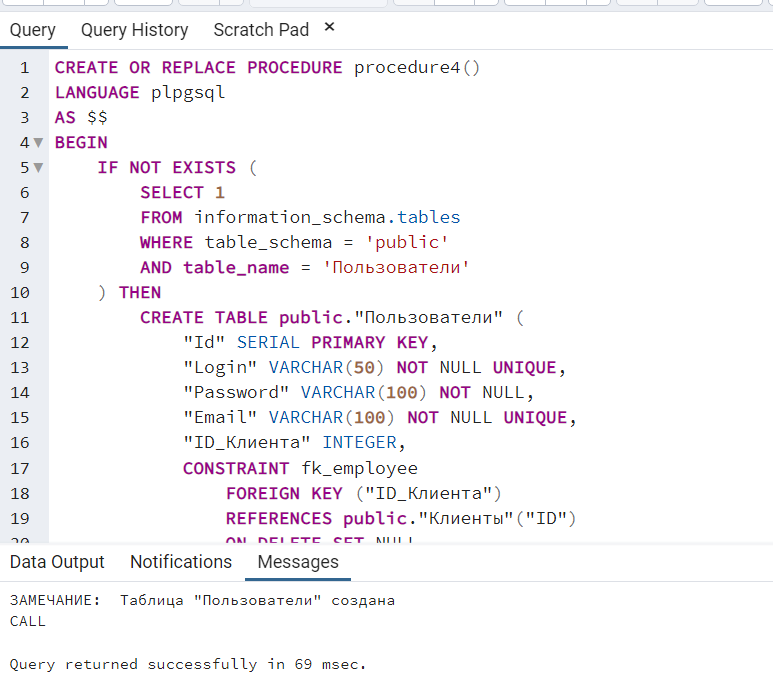
RAISE NOTICE 'Таблица "Пользователи" уже существует';

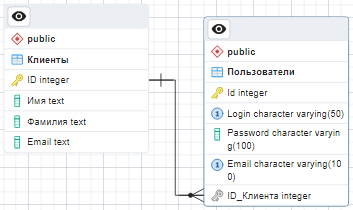
END IF;

END;

$$;

CALL procedure4();





1. Создайте хранимую процедуру генерирующую сложные пароли. Условие пароля: наличие служебных символов и цифр, наличие букв латинского алфавита обоих регистров, длинна пароля более 10 символов.

CREATE OR REPLACE PROCEDURE procedure5(INOUT gen\_password TEXT)

LANGUAGE plpgsql

AS $$

DECLARE

symbols TEXT := '!@#$%^&\*-\_abcdefghijklmnopqrstuvwxyz0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';

gen\_symbol TEXT;

low\_symbol BOOLEAN := FALSE;

high\_symbol BOOLEAN := FALSE;

num\_symbol BOOLEAN := FALSE;

spec\_symbol BOOLEAN := FALSE;

BEGIN

WHILE NOT (low\_symbol AND high\_symbol AND num\_symbol AND spec\_symbol) LOOP

gen\_password := '';

FOR i IN 1..11 LOOP

gen\_symbol := substr(symbols, ceil(random() \* length(symbols))::INT, 1);

gen\_password := gen\_password || gen\_symbol;

IF position(gen\_symbol IN 'abcdefghijklmnopqrstuvwxyz') > 0 THEN

low\_symbol := TRUE;

ELSIF position(gen\_symbol IN 'ABCDEFGHIJKLMNOPQRSTUVWXYZ') > 0 THEN

high\_symbol := TRUE;

ELSIF position(gen\_symbol IN '0123456789') > 0 THEN

num\_symbol := TRUE;

ELSIF position(gen\_symbol IN '!@#$%^&\*-\_') > 0 THEN

spec\_symbol := TRUE;

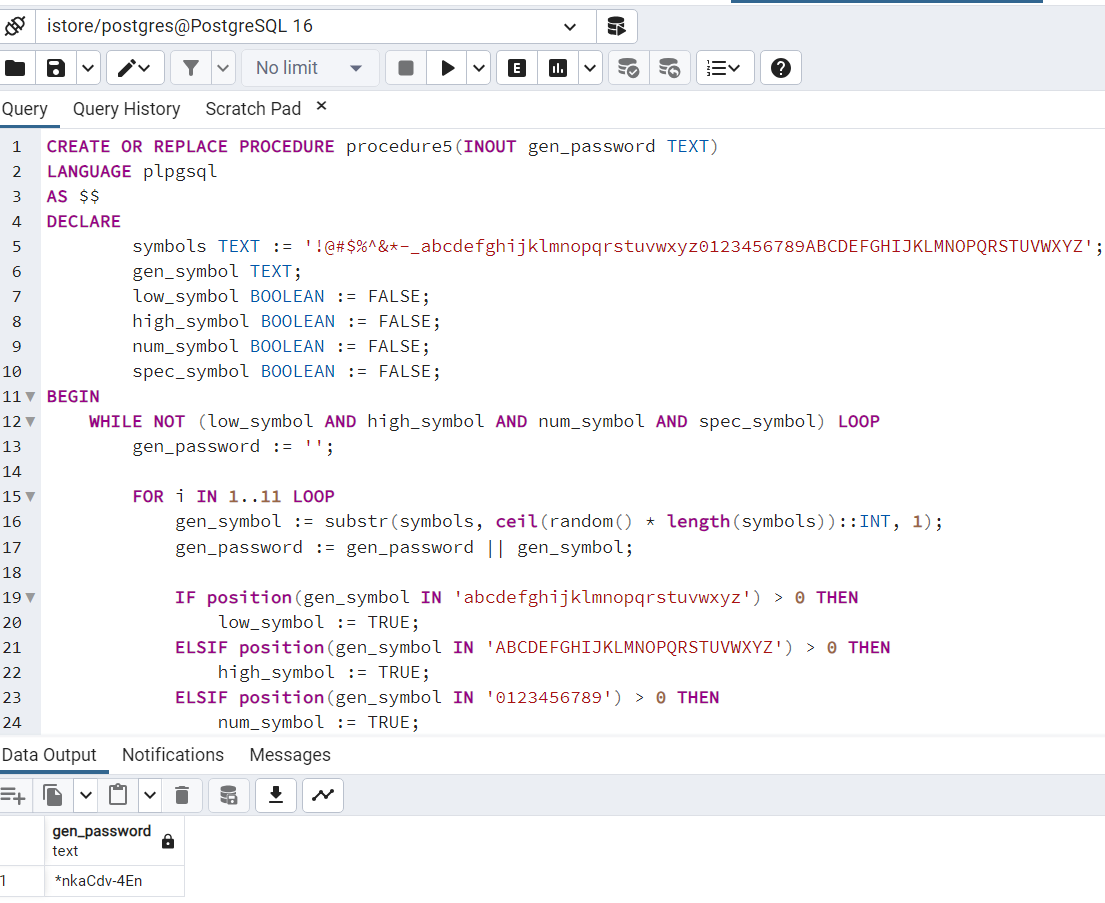
END IF;

END LOOP;

END LOOP;

END;

$$;

CALL procedure5(NULL); 

1. Создайте хранимую процедуру создающую на каждого клиента из таблицы Клиенты запись в таблице пользователь с автоматически сгенерированным паролем на основе хранимой процедуры из предыдущего задания и полем login полученным на основе инициалов клиента. Email копировать из таблицы клиенты.

CREATE OR REPLACE PROCEDURE procedure6()

LANGUAGE plpgsql

AS $$

DECLARE

client RECORD;

client\_password TEXT;

login TEXT;

BEGIN

-- Для каждого клиента

FOR client IN SELECT \* FROM "Клиенты" LOOP

-- 1. Генерируем пароль

CALL procedure5(client\_password);

-- 2. Создаем логин (первые буквы имени+фамилии + случайное число)

login := LOWER(

SUBSTRING(client."Имя" FROM 1 FOR 1) ||

SUBSTRING(client."Фамилия" FROM 1 FOR 1) ||

FLOOR(RANDOM() \* 90 + 10)::TEXT

);

-- 3. Добавляем пользователя

INSERT INTO "Пользователи" ("Login", "Password", "Email", "ID\_Клиента")

VALUES (login, client\_password, client."Email", client."Id")

ON CONFLICT DO NOTHING;

END LOOP;

END;

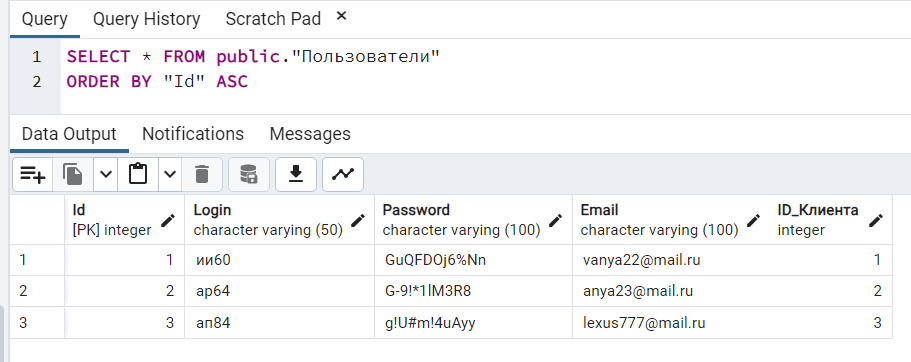
$$;

CALL procedure6();

SELECT \* FROM public."Пользователи"

ORDER BY "Id" ASC





1. Удалите все созданные вами процедуры.

DO $$

DECLARE

proc\_record RECORD;

BEGIN

FOR proc\_record IN

SELECT proname, oid FROM pg\_proc

WHERE pronamespace = (SELECT oid FROM pg\_namespace WHERE nspname = 'public')

AND proname NOT LIKE 'pg\_%'

LOOP

EXECUTE 'DROP PROCEDURE IF EXISTS ' || proc\_record.proname;

RAISE NOTICE 'Удалена процедура: %', proc\_record.proname;

END LOOP;

END $$;

