**1. Общие функции**

1.2 CREATE OR REPLACE FUNCTION random (a int, b int)

RETURNS int AS $$

DECLARE

res int;

BEGIN

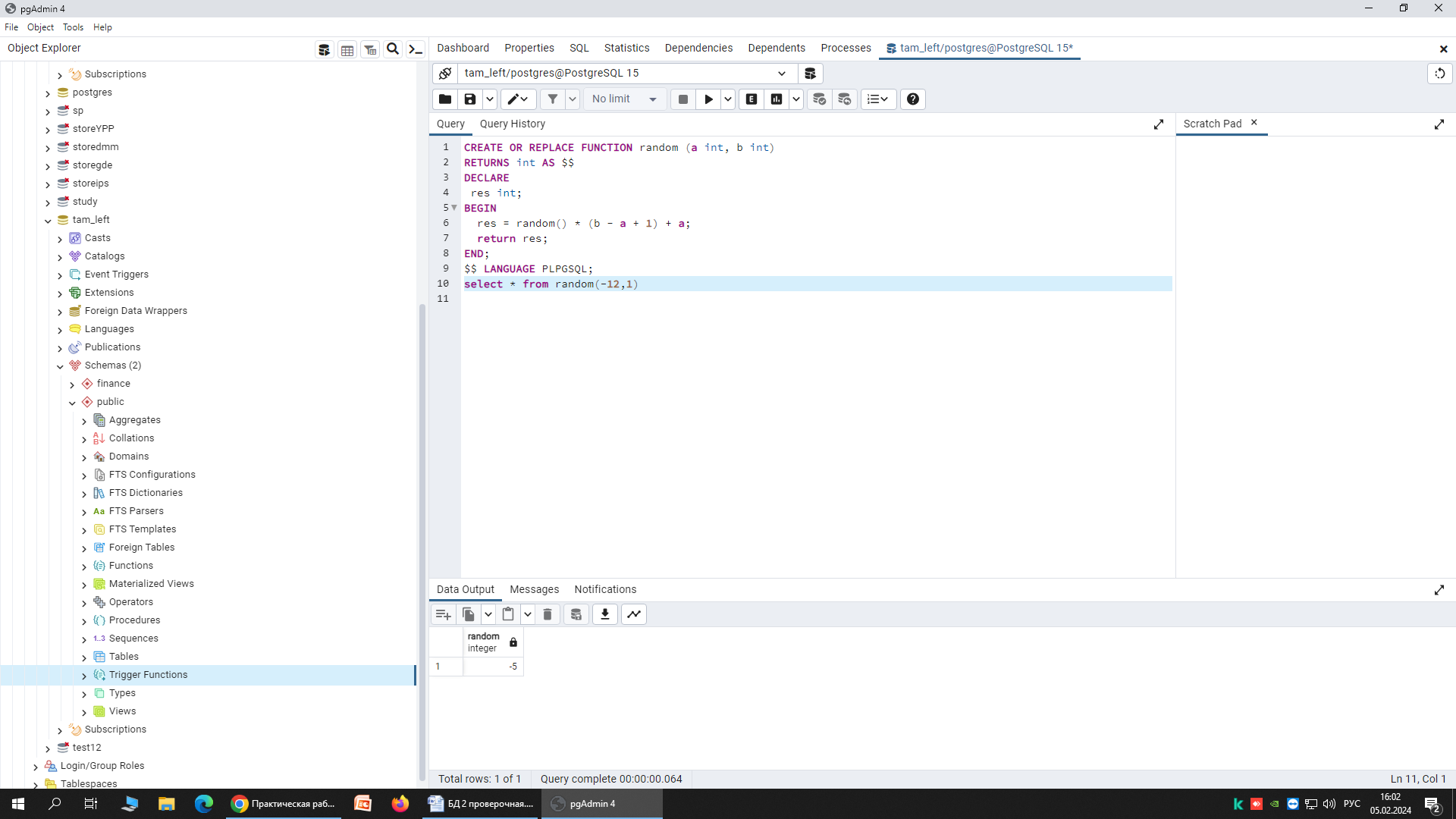
res = random() \* (b - a + 1) + a;

return res;

END;

$$ LANGUAGE PLPGSQL;

select \* from random(-12,1)



1.2 CREATE OR REPLACE FUNCTION random\_double (a int, b int, p int)

RETURNS decimal AS $$

DECLARE

num decimal;

res decimal;

BEGIN

num = random() \* (b - a)+a;

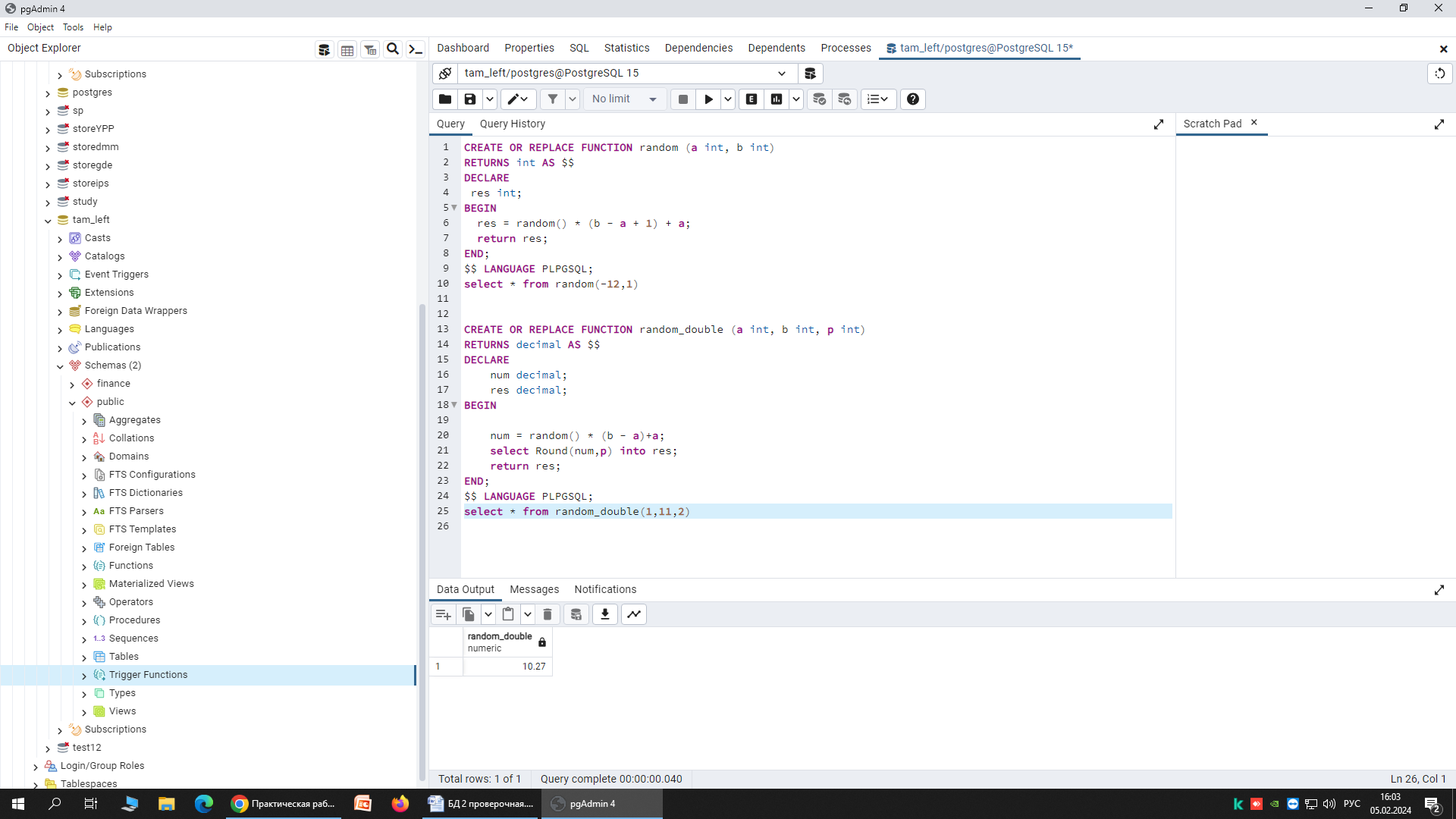
select Round(num,p) into res;

return res;

END;

$$ LANGUAGE PLPGSQL;

select \* from random\_double(1,11,2)



**2. Функции, работающие с массивами**

2.1 CREATE OR REPLACE FUNCTION mult\_arr(arr numeric[], k int)

RETURNS decimal[] AS $$

DECLARE

x int;

BEGIN

FOREACH x IN ARRAY arr

LOOP

arr[x]:=arr[x]\*k;

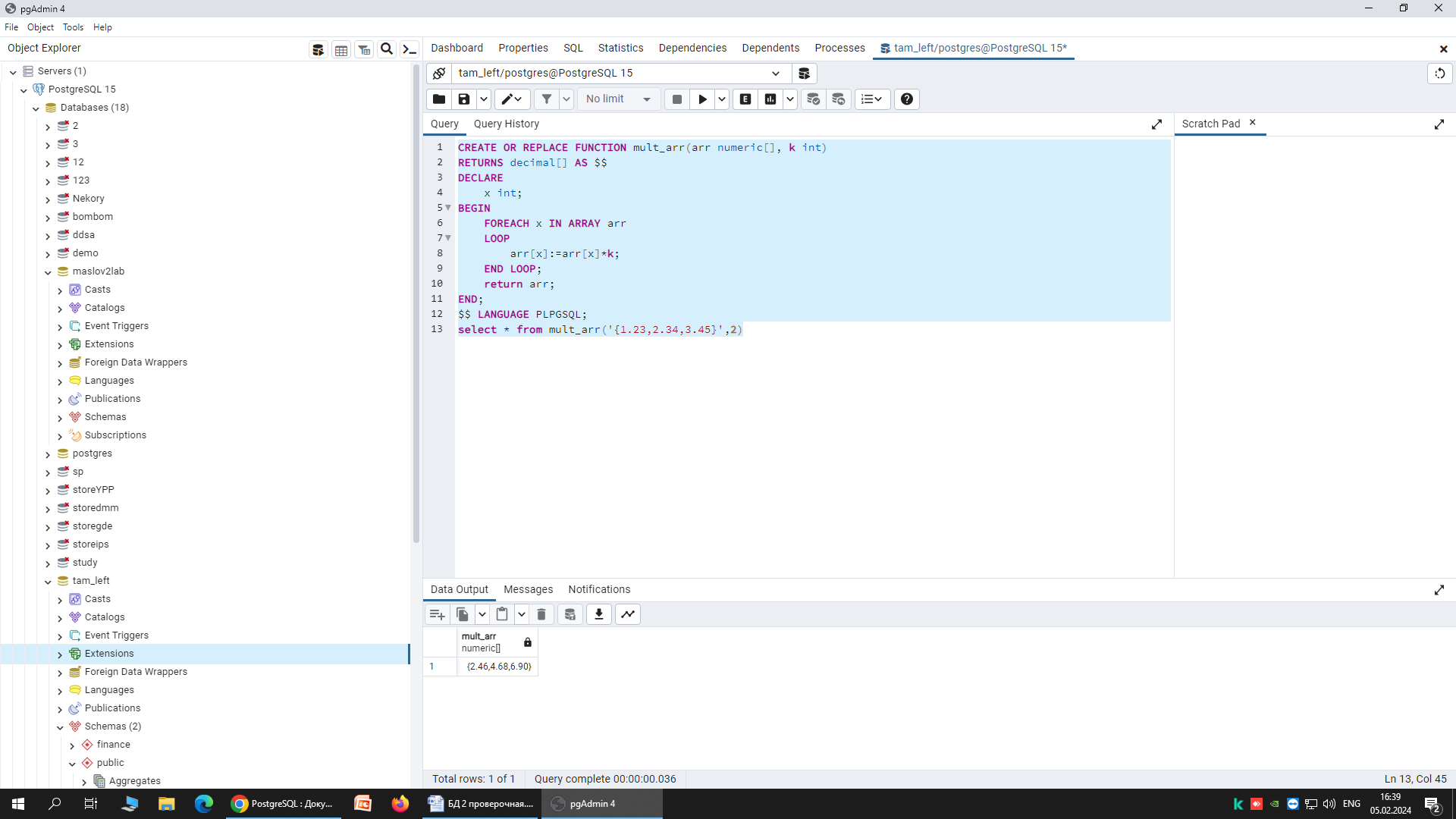
END LOOP;

return arr;

END;

$$ LANGUAGE PLPGSQL;

select \* from mult\_arr('{1.23,2.34,3.45}',2)



2.2 create or replace function get\_arr(s text, n numeric, i integer)

returns text[] as $$

declare

arr text[];

begin

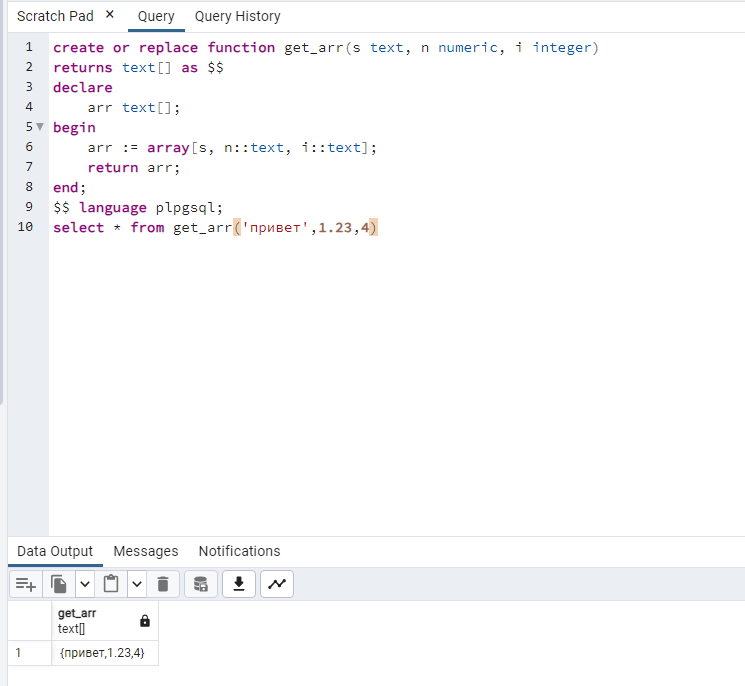
arr := array[s, n::text, i::text];

return arr;

end;

$$ language plpgsql;

select \* from get\_arr('привет',1.23,4)



**3. Функции с произвольным числом аргументов**

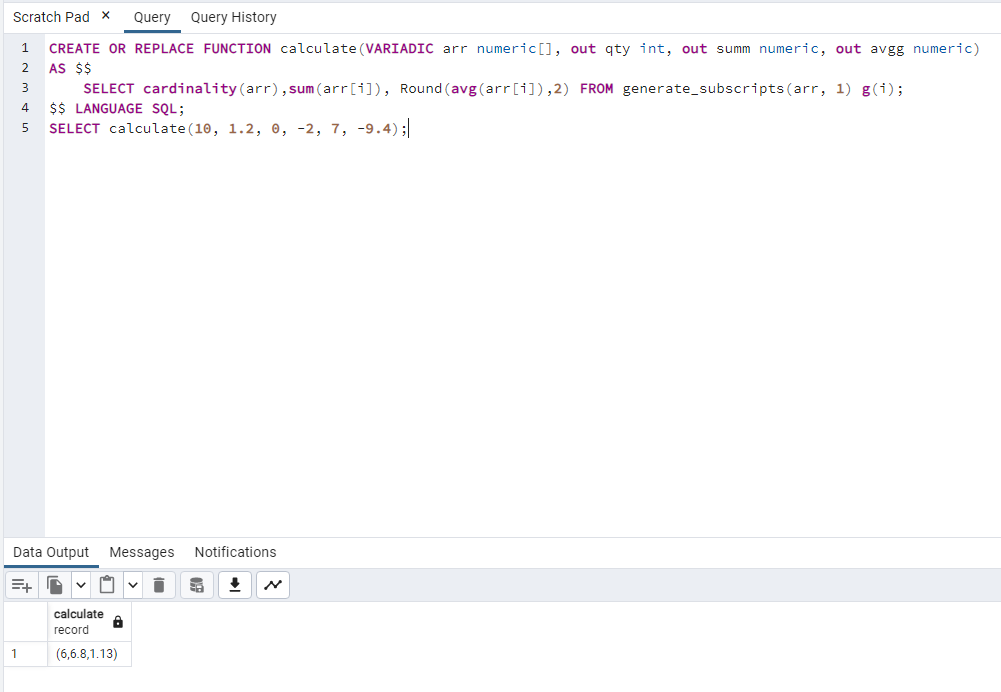
3.1 CREATE OR REPLACE FUNCTION calculate(VARIADIC arr numeric[], out qty int, out summ numeric, out avgg numeric)

AS $$

SELECT cardinality(arr),sum(arr[i]), Round(avg(arr[i]),2) FROM generate\_subscripts(arr, 1) g(i);

$$ LANGUAGE SQL;

SELECT calculate(10, 1.2, 0, -2, 7, -9.4);



**4. Функции и процедуры для выполнения CRUD-операций (общая база данных)**

4.1 create or replace procedure pr\_add\_category(name varchar(50))

language plpgsql as $$

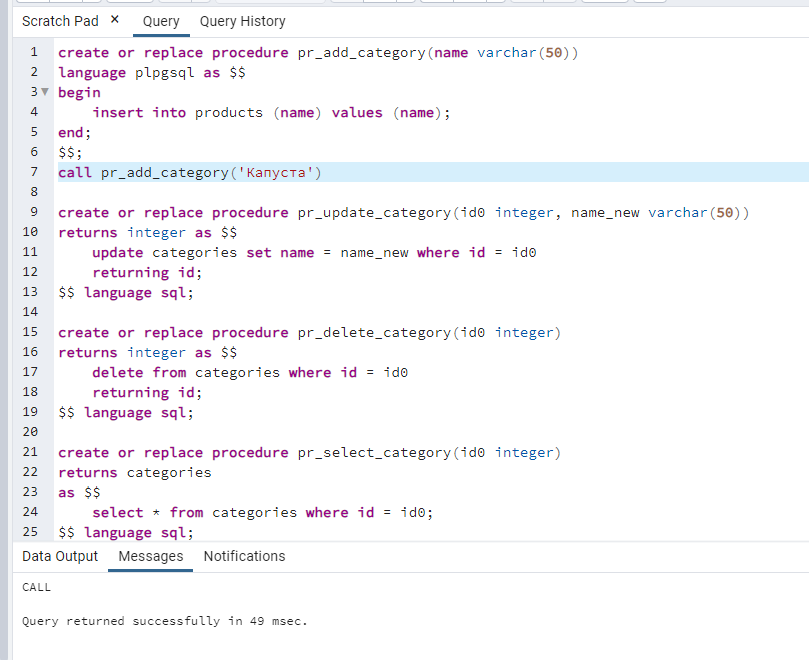
begin

insert into products (name) values (name);

end;

$$;

call pr\_add\_category('Капуста')



create or replace procedure pr\_update\_category(cur\_id integer, name\_new varchar(50))

language plpgsql as $$

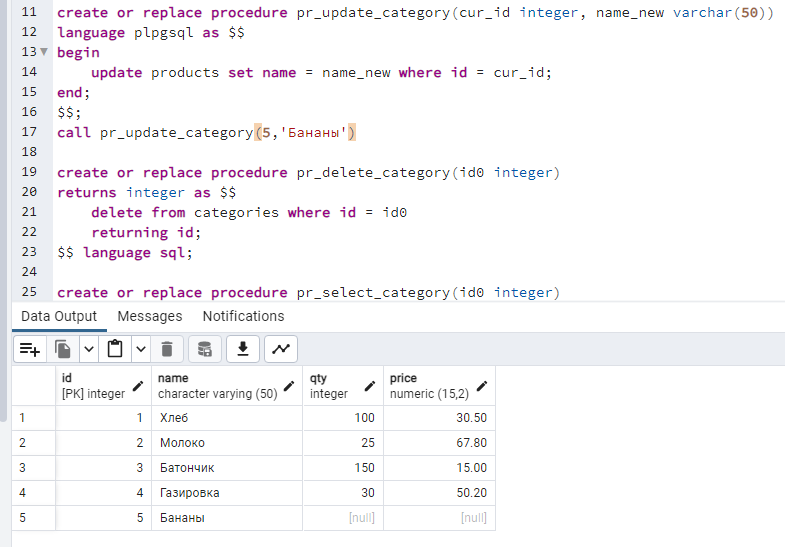
begin

update products set name = name\_new where id = cur\_id;

end;

$$;

call pr\_update\_category(5,'Бананы')



create or replace procedure pr\_delete\_category(cur\_id integer)

language plpgsql as $$

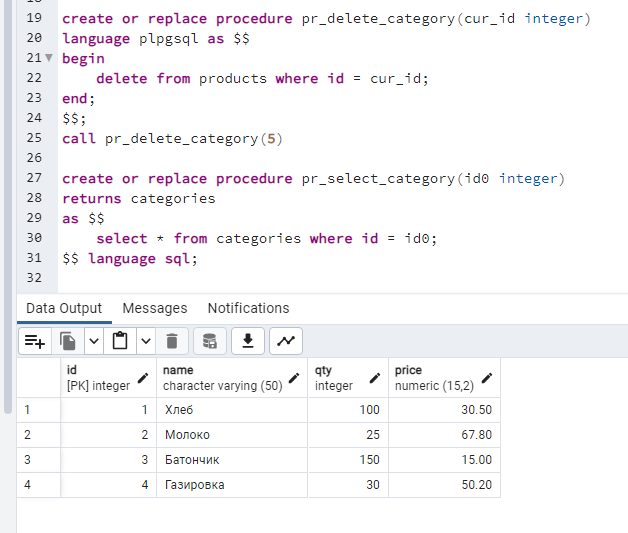
begin

delete from products where id = cur\_id;

end;

$$;

call pr\_delete\_category(5)



create or replace procedure pr\_select\_category(cur\_id integer)

language plpgsql as $$

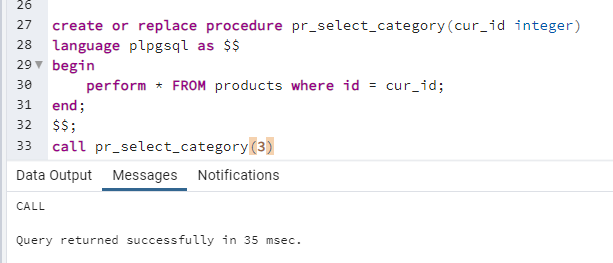
begin

perform \* FROM products where id = cur\_id;

end;

$$;

call pr\_select\_category(3)



4.2 create or replace function f\_add\_category(name varchar(50))

returns varchar(50) as $$

begin

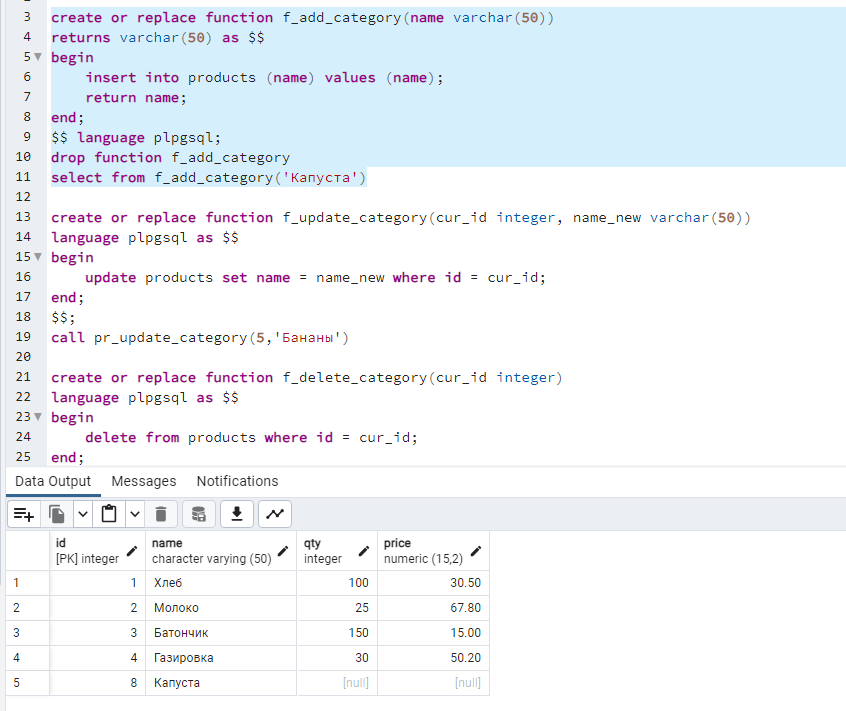
insert into products (name) values (name);

return name;

end;

$$ language plpgsql;

select from f\_add\_category('Капуста')



create or replace function f\_update\_category(cur\_id integer, name\_new varchar(50))

returns boolean as $$

begin

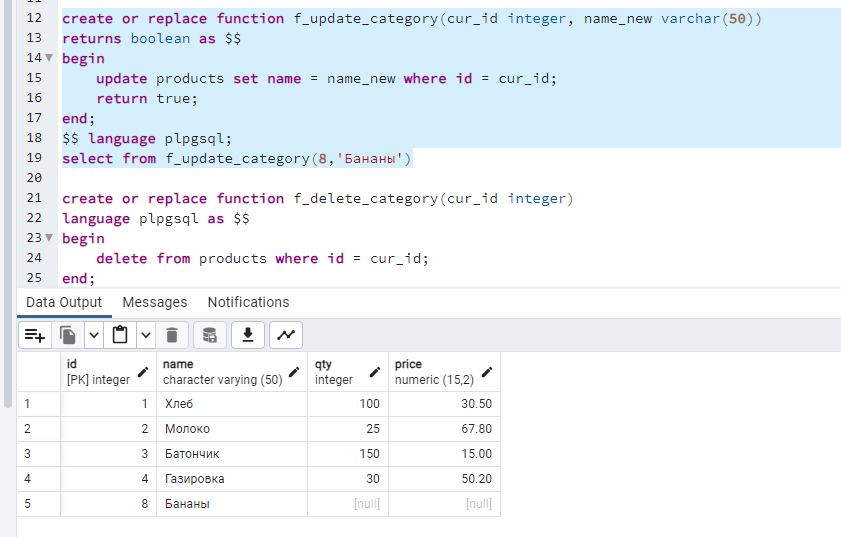
update products set name = name\_new where id = cur\_id;

return true;

end;

$$ language plpgsql;

select from f\_update\_category(8,'Бананы')



create or replace function f\_delete\_category(cur\_id integer)

returns boolean as $$

begin

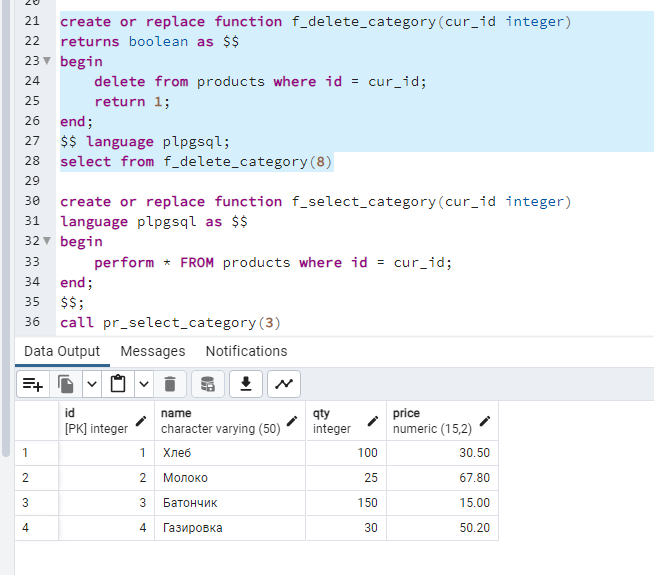
delete from products where id = cur\_id;

return 1;

end;

$$ language plpgsql;

select from f\_delete\_category(8)



create or replace function f\_select\_category(cur\_id integer)

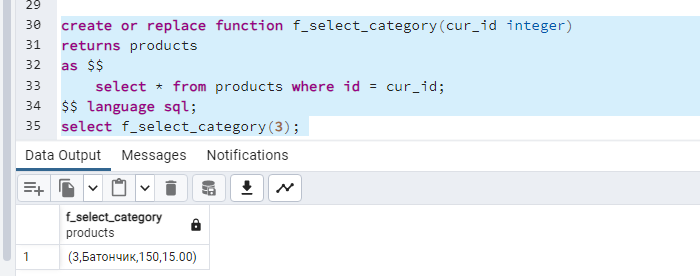
returns products

as $$

select \* from products where id = cur\_id;

$$ language sql;

select f\_select\_category(3);



1. Добавляет новую запись
2. Использование if not exist
3. 1
4. 1
5. select f\_select\_category(3); - вывод массивом

select \* from f\_select\_category(3); - вывод таблицей

select (f\_select\_category(3)).id, (f\_select\_category(3)).name; - вывод id и name

1. Вызвать можно, функция вернёт значение Boolean

**5. Функции, возвращающие множество значений и таблицу**

5.1 create or replace function no\_categories()

returns setof text as $$

begin

return query select category\_name::text from categories

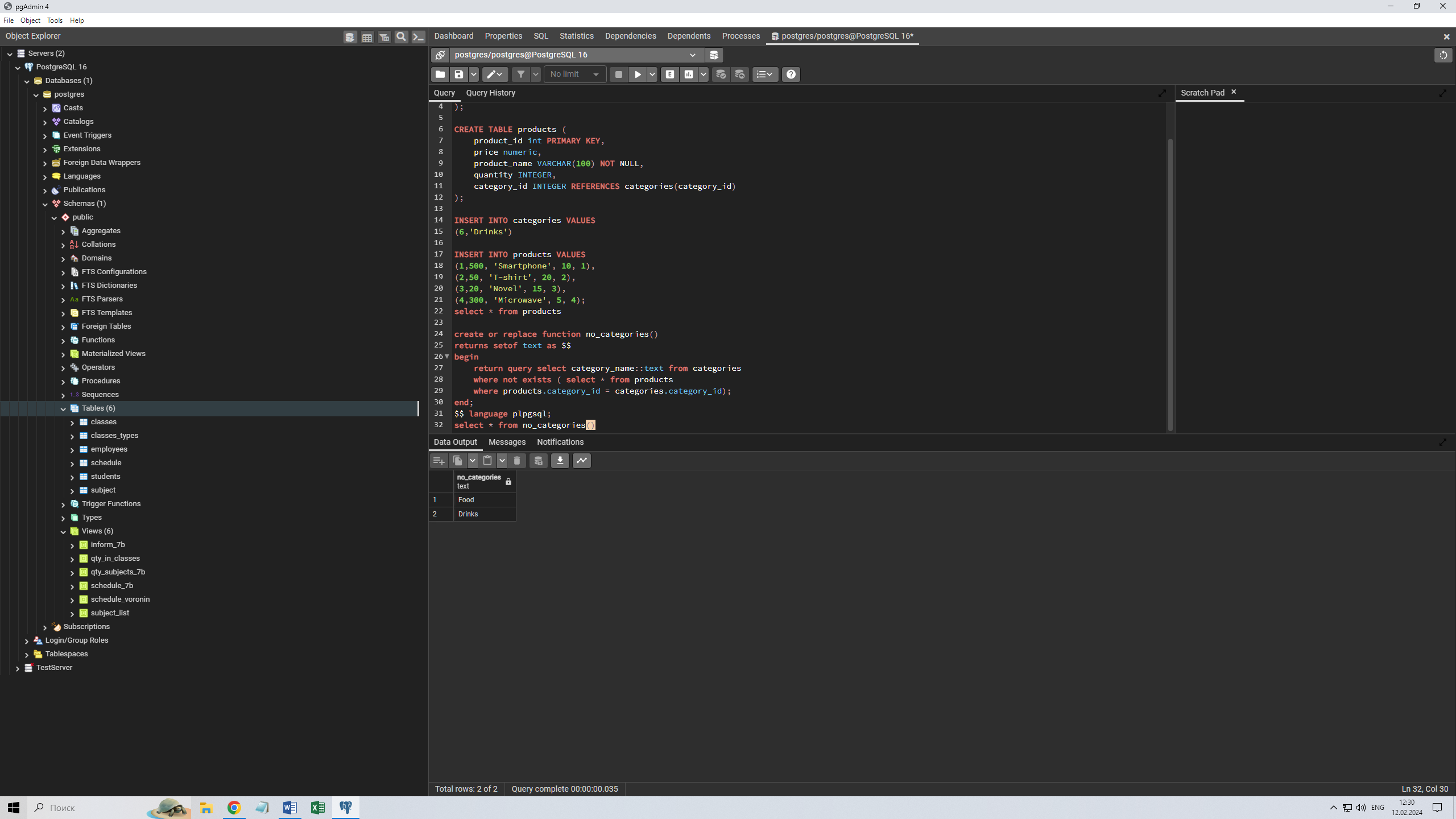
where not exists ( select \* from products

where products.category\_id = categories.category\_id);

end;

$$ language plpgsql;

select \* from no\_categories()



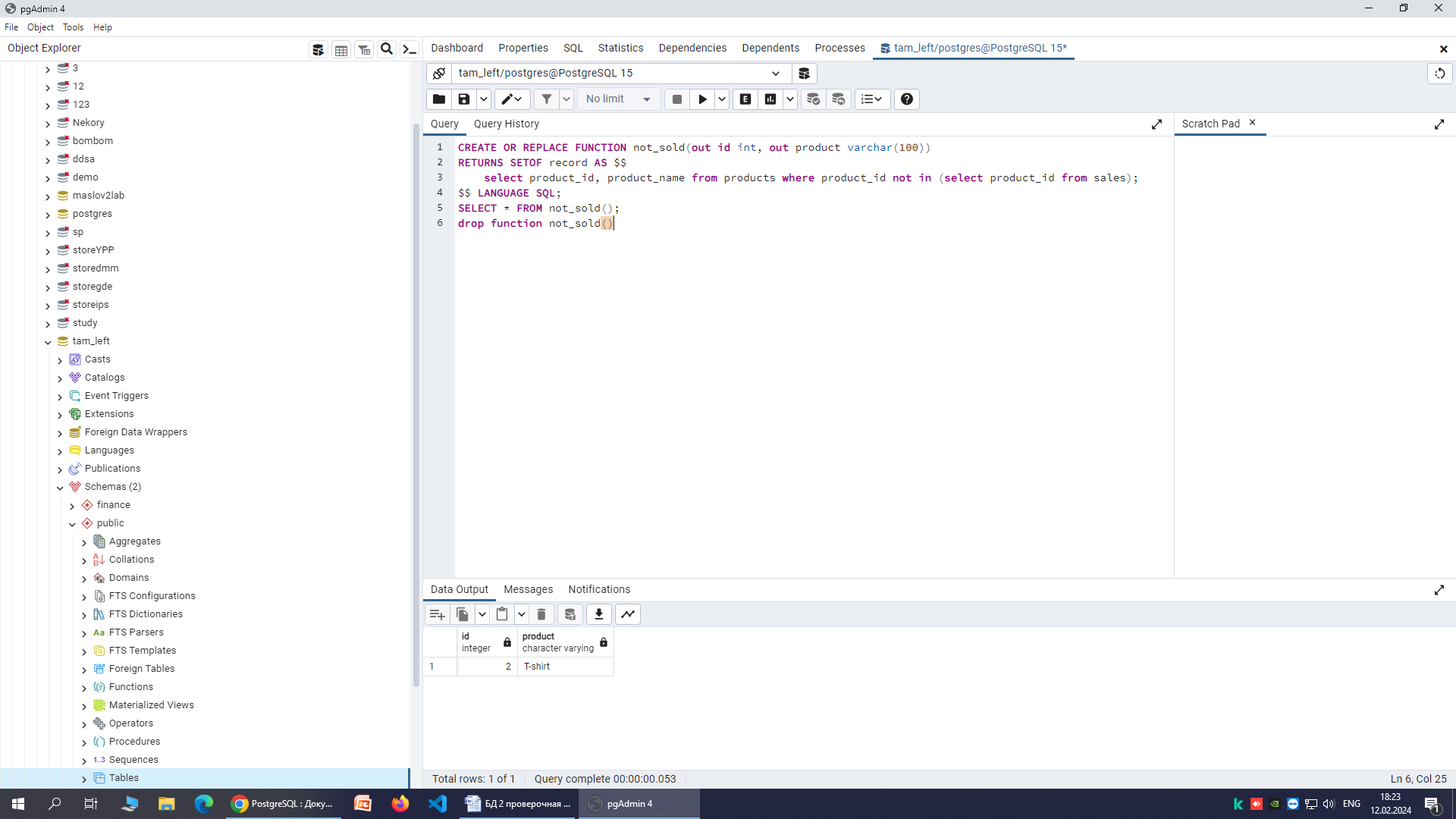
5.2 CREATE OR REPLACE FUNCTION not\_sold(out id int, out product varchar(100))

RETURNS SETOF record AS $$

select product\_id, product\_name from products where product\_id not in (select product\_id from sales);

$$ LANGUAGE SQL;

SELECT \* FROM not\_sold();



5.3 CREATE OR REPLACE FUNCTION orders\_stat(out countt bigint, out summ numeric(15,2), out maxx numeric(15,2),

out minn numeric(15,2), out avgg numeric(15,2))

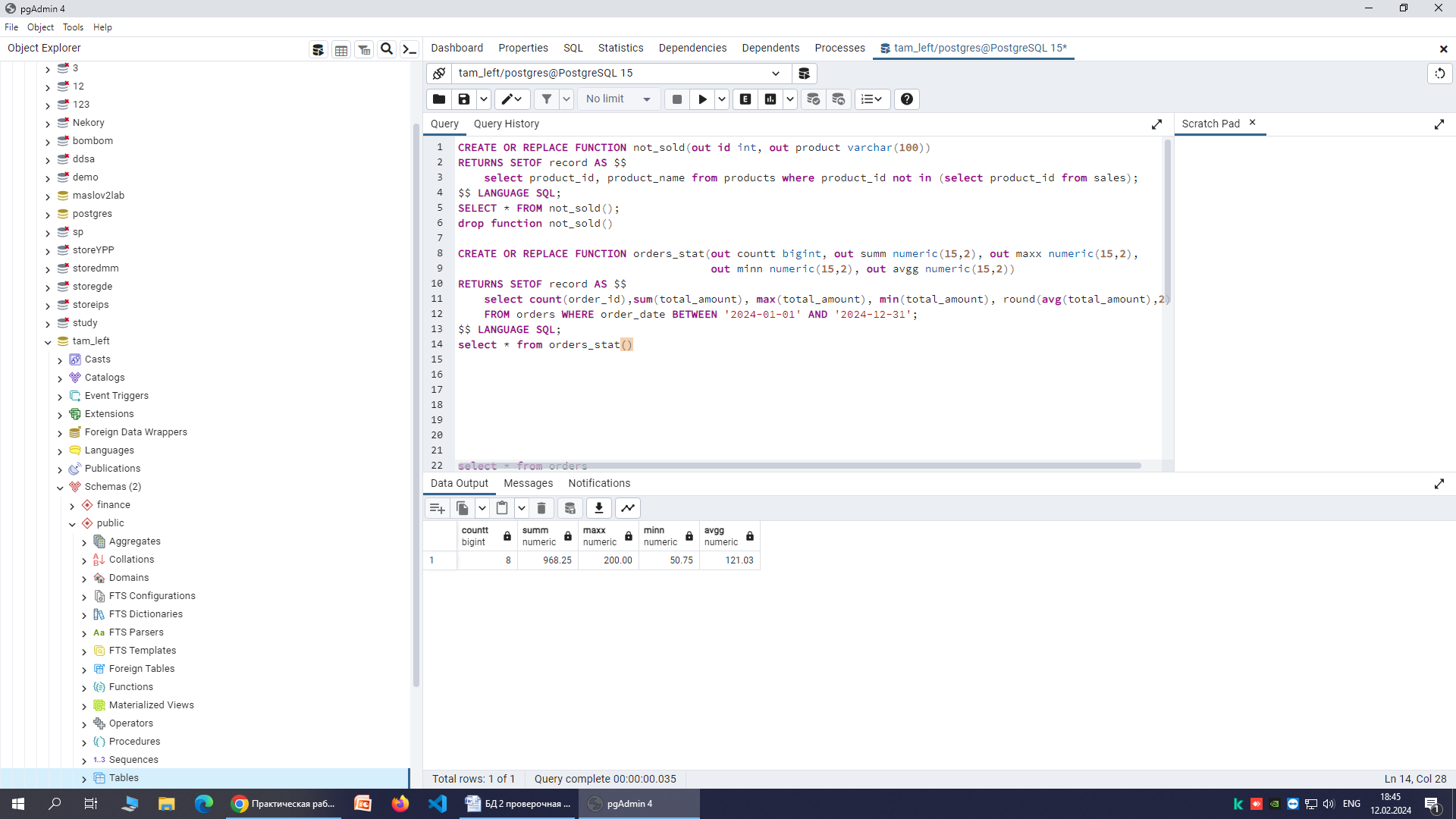
RETURNS SETOF record AS $$

select count(order\_id),sum(total\_amount), max(total\_amount), min(total\_amount), round(avg(total\_amount),2)

FROM orders WHERE order\_date BETWEEN '2024-01-01' AND '2024-12-31';

$$ LANGUAGE SQL;

select \* from orders\_stat()



5.4 CREATE or replace FUNCTION client\_info(start\_date DATE, end\_date DATE)

RETURNS TABLE (l\_name VARCHAR(50), f\_name VARCHAR(50), s\_name VARCHAR(50), rtng int, count\_orders bigint, last\_order DATE)

AS $$

BEGIN

RETURN QUERY SELECT customers.second\_name, customers.first\_name, customers.patronymic, customers.rating,

COUNT(orders.order\_id) AS count\_orders, MAX(orders.date) AS last\_order

FROM customers

JOIN orders ON customers.id = orders.customer\_id

WHERE orders.status NOT IN ('Черновик', 'Отменен')

AND orders.date BETWEEN start\_date AND end\_date

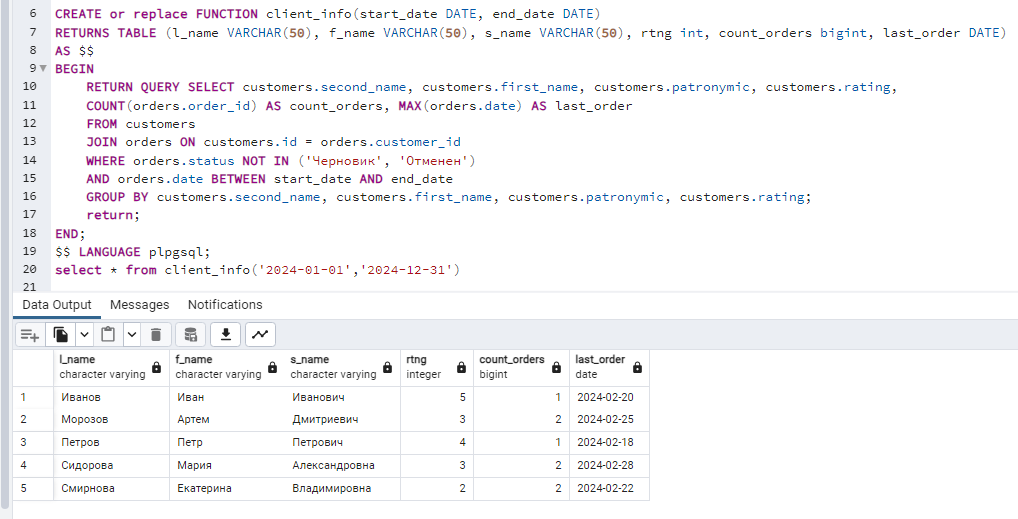
GROUP BY customers.second\_name, customers.first\_name, customers.patronymic, customers.rating;

return;

END;

$$ LANGUAGE plpgsql;

select \* from client\_info('2024-01-01','2024-12-31')



CREATE TABLE customers (

id int primary Key ,

second\_name VARCHAR(50),

first\_name VARCHAR(50),

patronymic VARCHAR(50),

rating INT

);

INSERT INTO customers VALUES

(1,'Иванов', 'Иван', 'Иванович', 5),

(2,'Петров', 'Петр', 'Петрович', 4),

(3,'Сидорова', 'Мария', 'Александровна', 3),

(4,'Козлов', 'Алексей', 'Николаевич', 4),

(5,'Смирнова', 'Екатерина', 'Владимировна', 2),

(6,'Николаев', 'Андрей', 'Петрович', 4),

(7,'Григорьева', 'Ольга', 'Игоревна', 5),

(8,'Морозов', 'Артем', 'Дмитриевич', 3);

CREATE TABLE products (

id INT primary key,

name VARCHAR(50),

price DECIMAL(10, 2),

qty INT

);

INSERT INTO products (id, name, price, qty) VALUES

(1, 'Книга', 25.00, 100),

(2, 'Флешка', 10.50, 50),

(3, 'Футболка', 15.75, 75),

(4, 'Мышка', 20.25, 60),

(5, 'Чехол для телефона', 8.00, 40),

(6, 'Наушники', 30.50, 45),

(7, 'Фоторамка', 5.25, 30),

(8, 'Зеркало', 12.00, 20);

CREATE TABLE orders (

order\_id INT,

product\_id INT references products(id),

customer\_id int references customers(id),

date DATE,

amount DECIMAL(10, 2),

status VARCHAR(20)

);

INSERT INTO orders VALUES

(1, 2,8, '2024-02-10', 21.00,'Новый'),

(2, 4,5, '2024-02-12', 40.50,'Доставлен'),

(3, 6,3, '2024-02-15', 61.00,'В работе'),

(4, 1,2, '2024-02-18', 30.00,'Выдан'),

(5, 3,1, '2024-02-20', 22.50,'Новый'),

(6, 7,5, '2024-02-22', 7.50,'Выдан'),

(7, 5,8, '2024-02-25', 16.00,'В работе'),

(8, 2,3, '2024-02-28', 10.50,'Новый');