**1. Общие функции**

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

RETURNS int AS $$

DECLARE

res int;

BEGIN

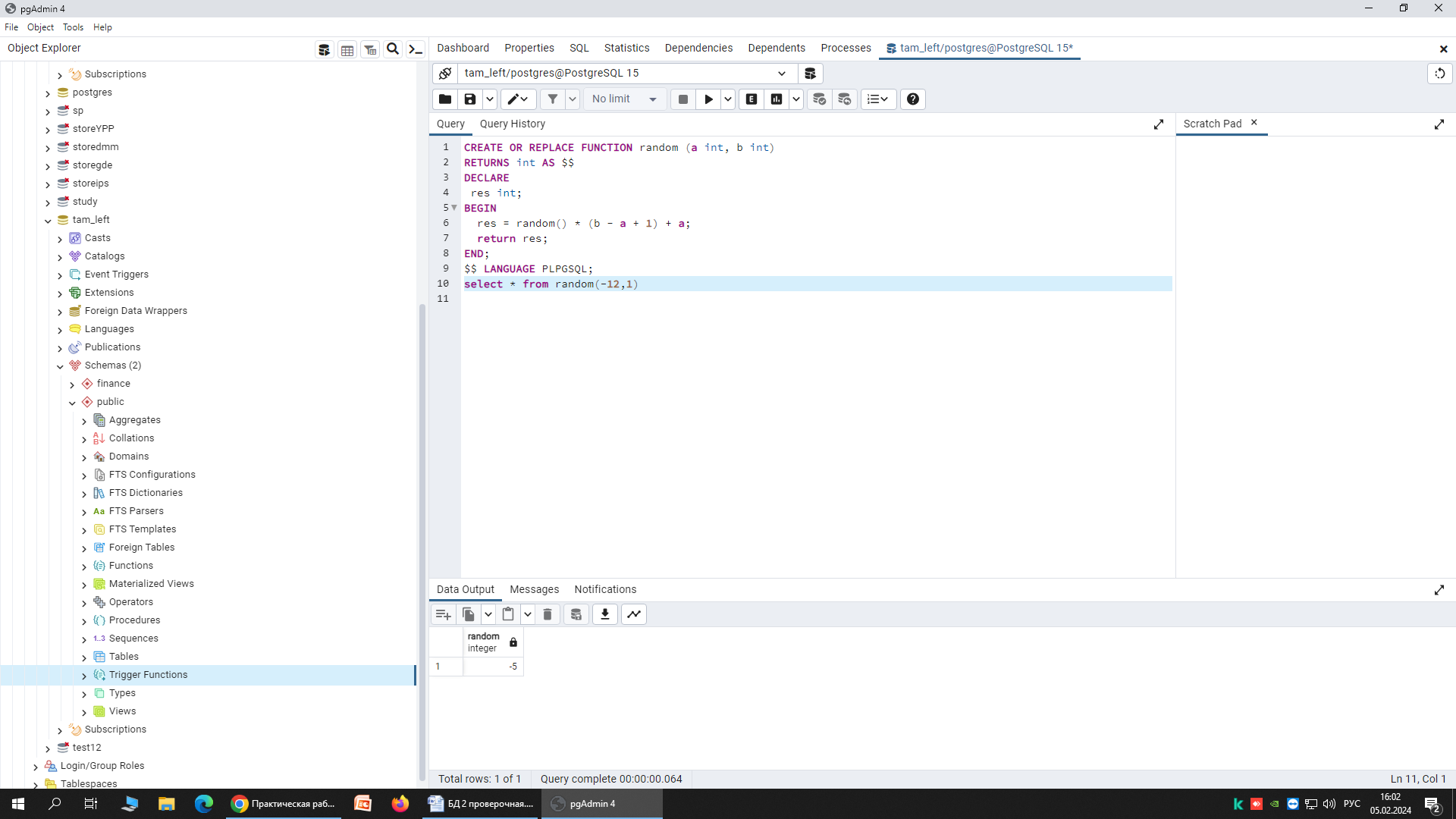
res = random() \* (b - a + 1) + a;

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;

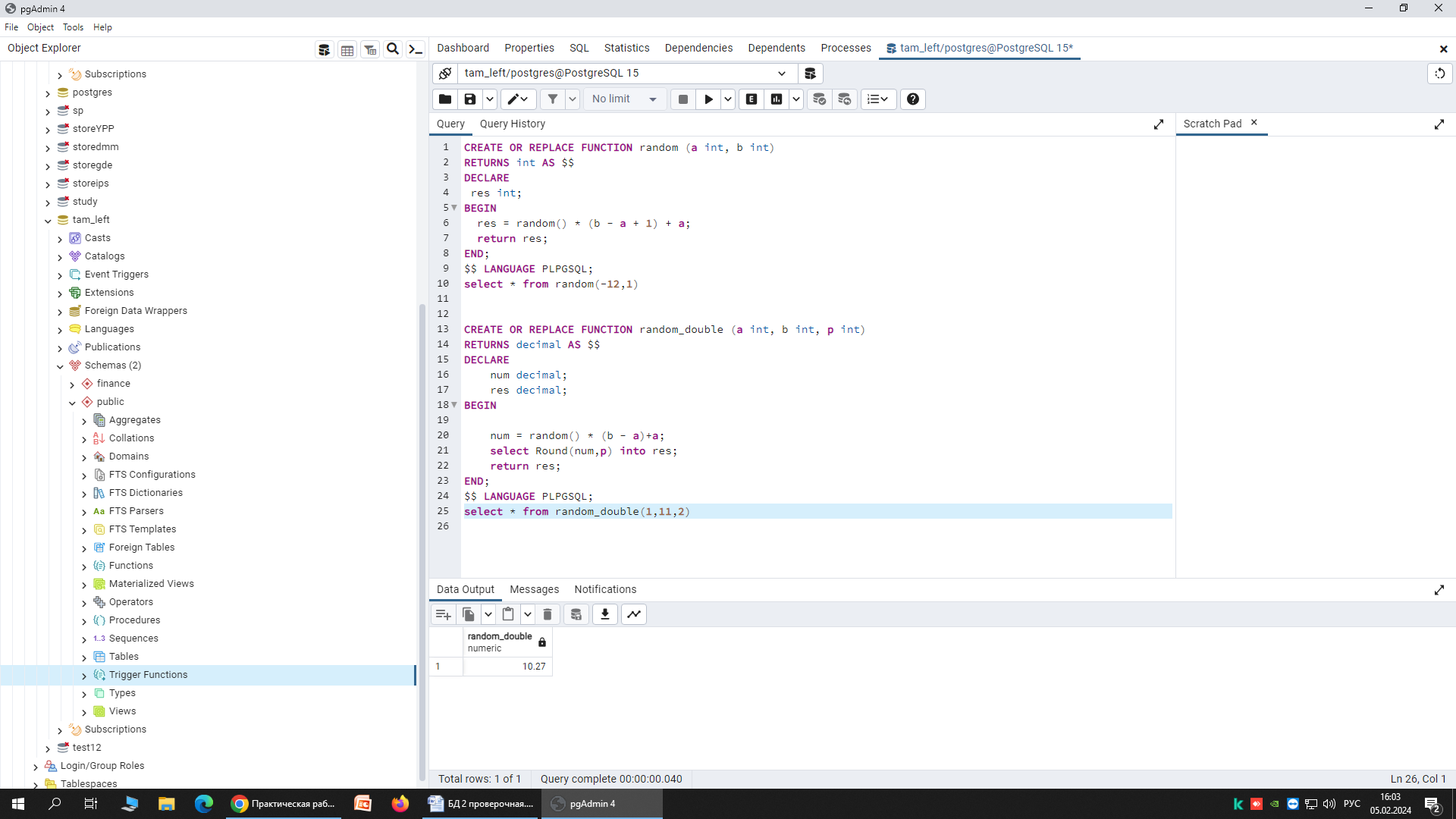
select Round(num,p) into res;

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;

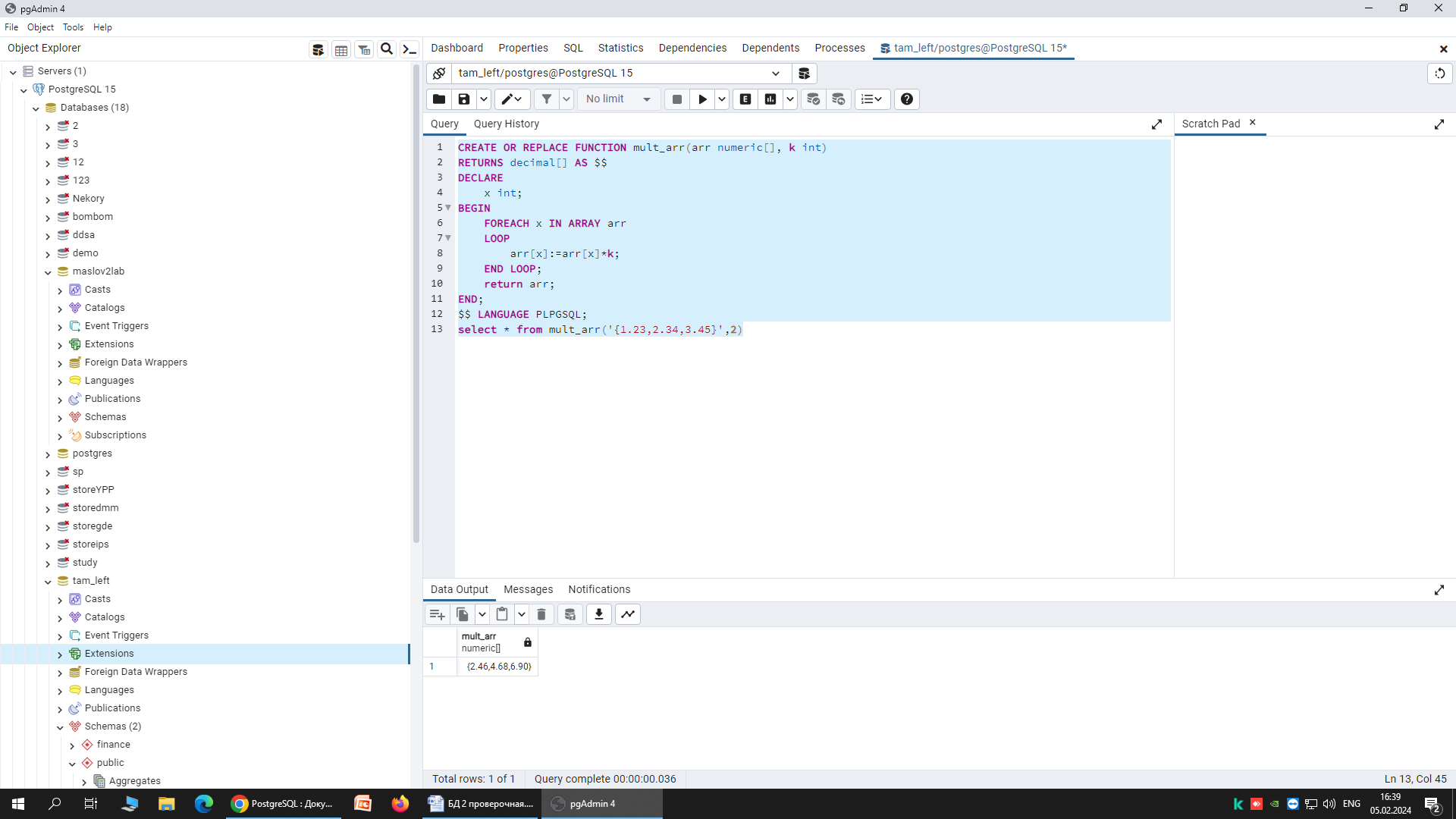
END LOOP;

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

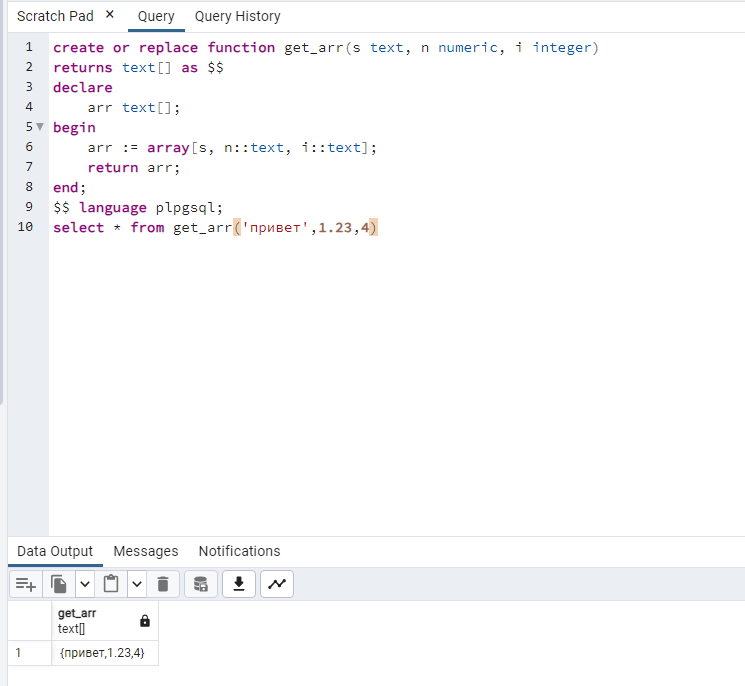
arr := array[s, n::text, i::text];

return arr;

end;

$$ language plpgsql;

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



**3. Функции с произвольным числом аргументов**

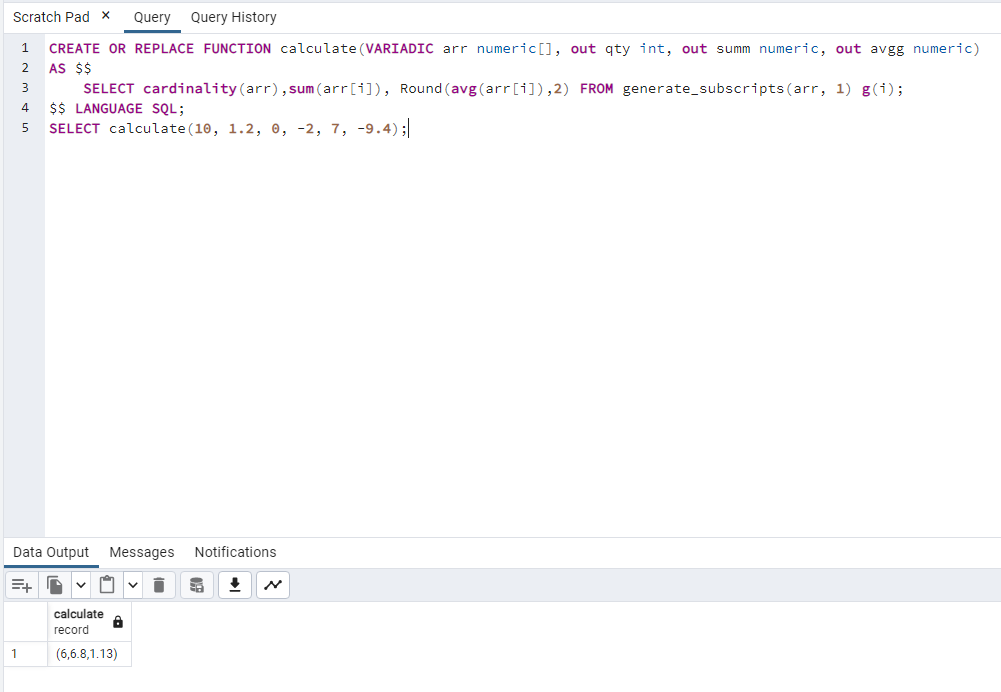
3.1 CREATE OR REPLACE FUNCTION calculate(VARIADIC arr numeric[], out qty int, out summ numeric, out avgg numeric)

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 $$

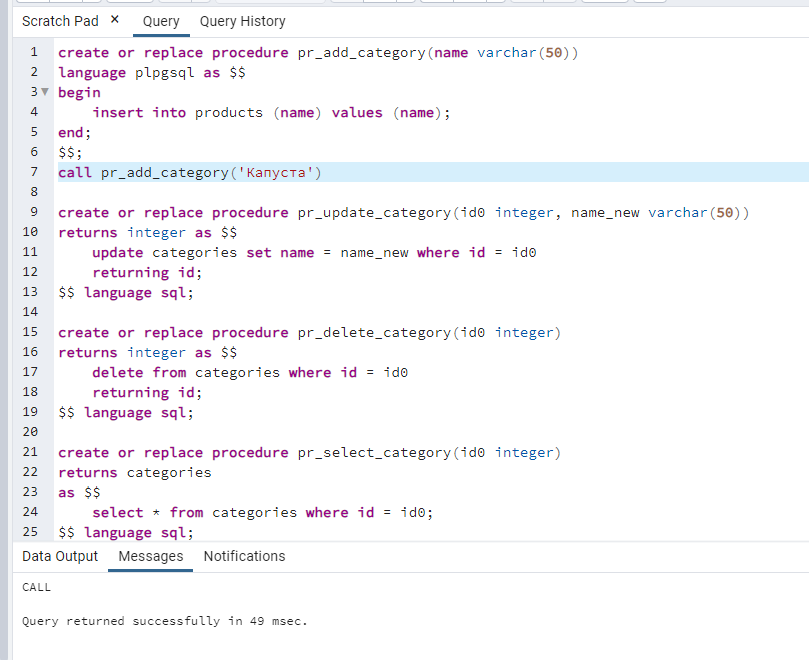
begin

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 $$

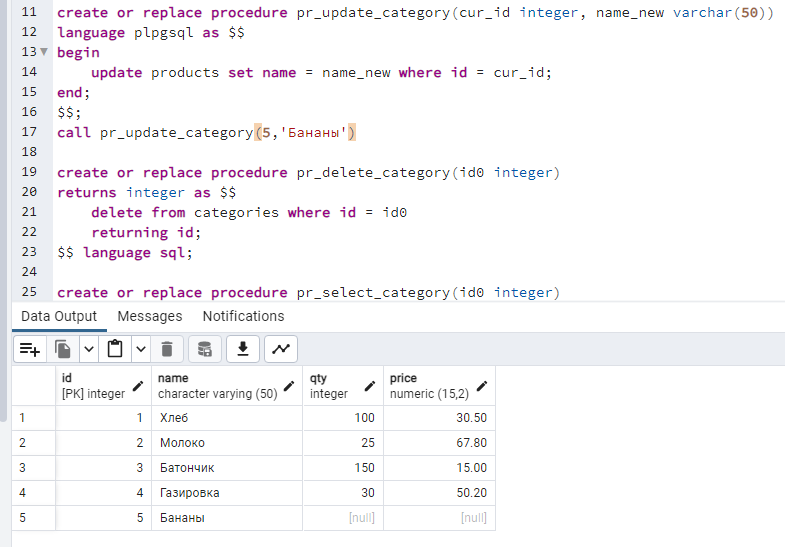
begin

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 $$

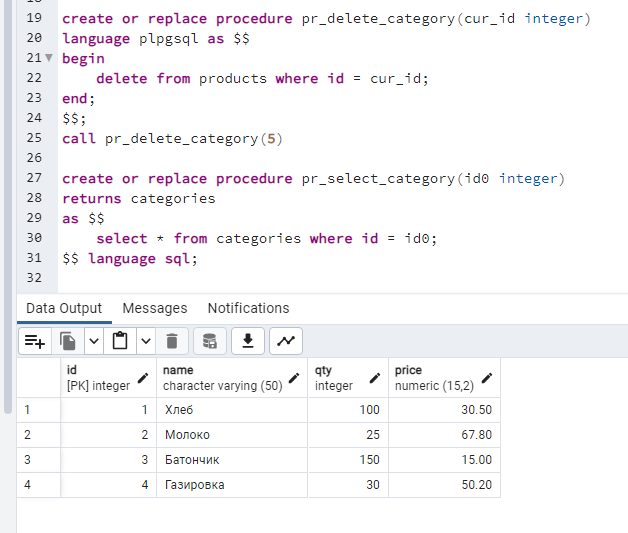
begin

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 $$

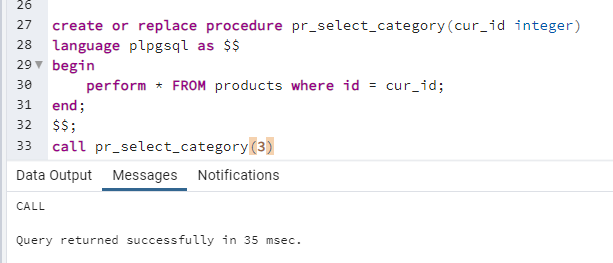
begin

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

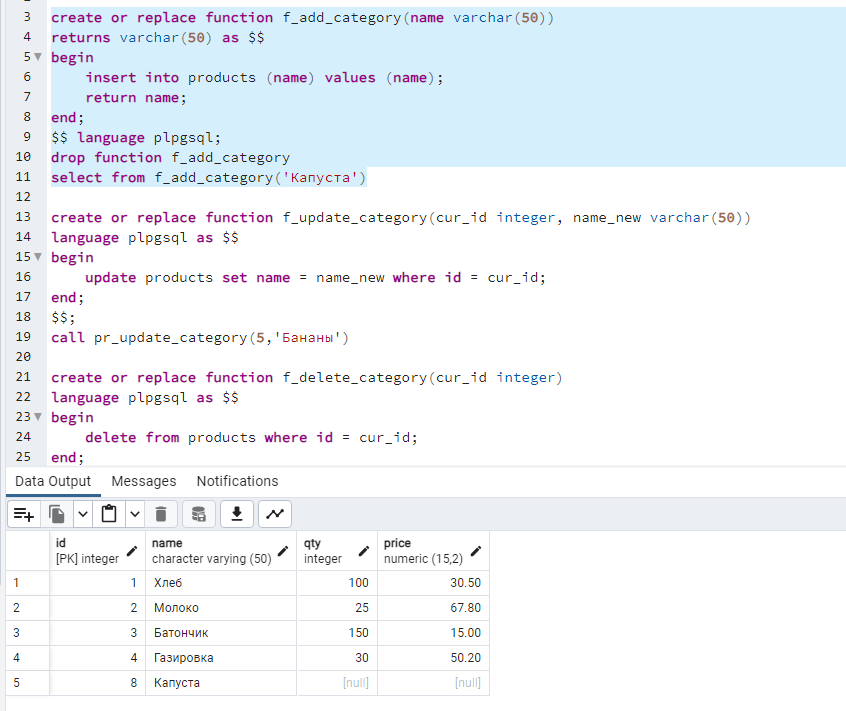
insert into products (name) values (name);

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

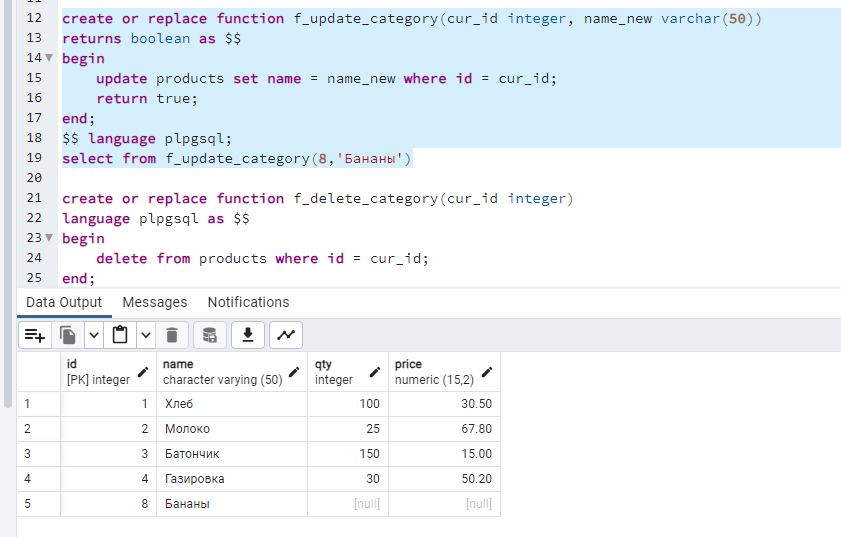
update products set name = name\_new where id = cur\_id;

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

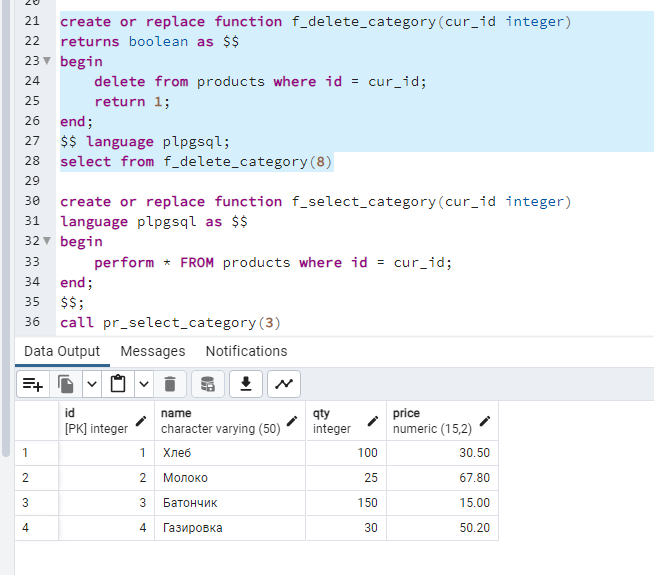
delete from products where id = cur\_id;

return 1;

end;

$$ language plpgsql;

select from f\_delete\_category(8)



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

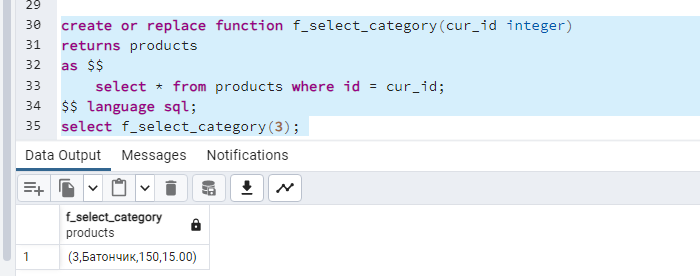
returns products

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

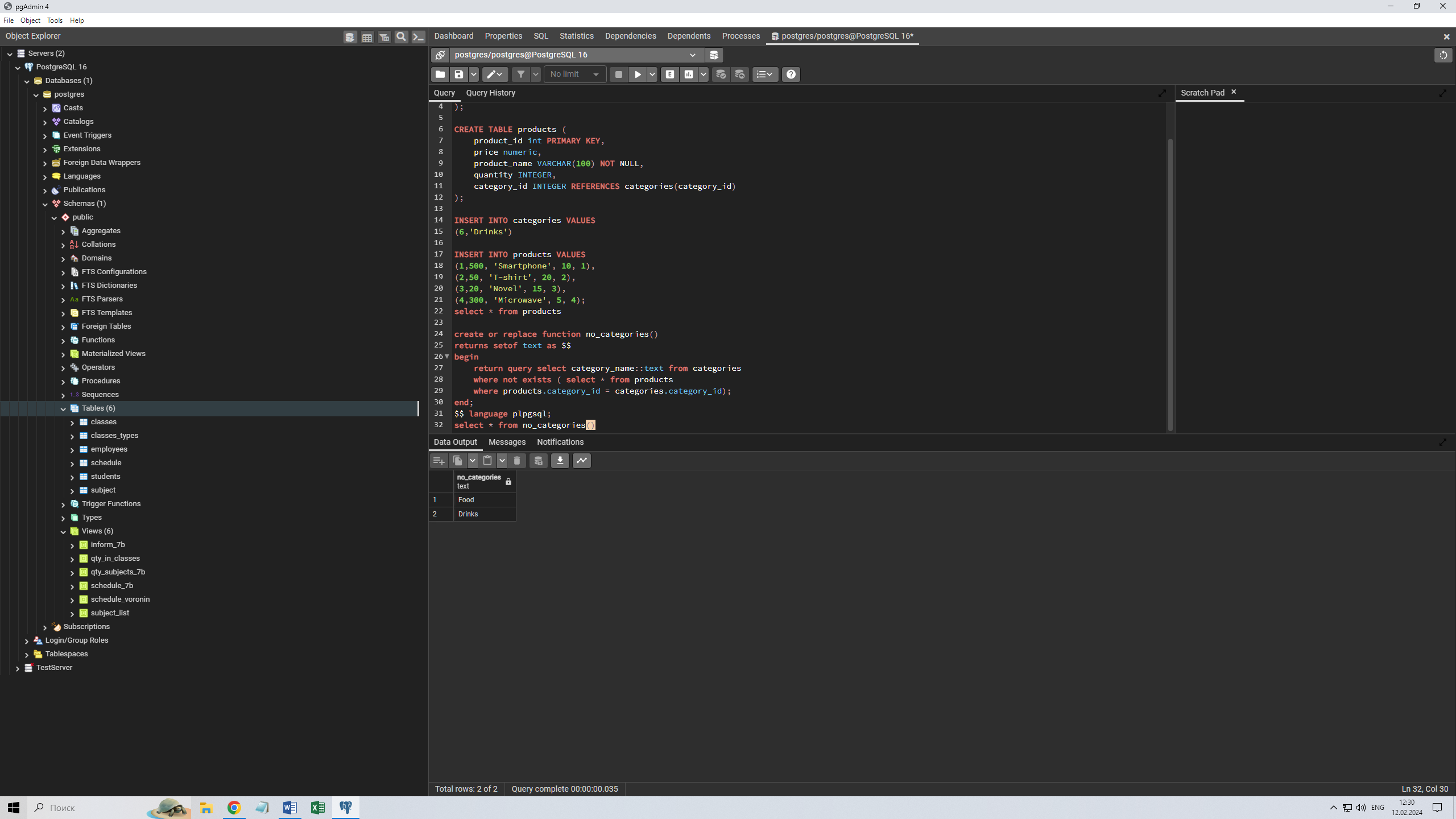
where not exists ( select \* from products

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

end;

$$ language plpgsql;

select \* from no\_categories()



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

RETURNS SETOF record AS $$

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

$$ LANGUAGE SQL;

SELECT \* FROM not\_sold();

CREATE TABLE sales (

sale\_id int PRIMARY KEY,

product\_id int,

quantity\_sold INT

);

INSERT INTO sales VALUES

(1,1,100),

(2,3,48),

(3,4,91);

CREATE TABLE products (

product\_id int PRIMARY KEY,

price numeric,

product\_name VARCHAR(100) NOT NULL,

quantity INTEGER,

category\_id INTEGER REFERENCES categories(category\_id)

);

INSERT INTO categories VALUES

(1, 'Electronics'),

(2, 'Clothing'),

(3, 'Books'),

(4, 'Home Appliances'),

(5, 'Food');

INSERT INTO products VALUES

(1, 500, 'Smartphone', 10, 1),

(2, 50, 'T-shirt', 20, 2),

(3, 20, 'Novel', 15, 3),

(4, 300, 'Microwave', 5, 4);