МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники



**Курсовая работа**

**по дисциплине:** «Базы данных»

**На тему:** «Разработка базы данных «Детские товары»»

Выполнил: Проверил:

Студент гр. «АВТ-018» к.т.н., доцент

Наумова Д.К.Трошина Г.В.

«\_\_\_» \_\_\_\_\_\_ 20\_\_г.«\_\_\_» \_\_\_\_\_\_ 20\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2022 г.

Содержание

[1. Задание…………………………………………………………………….…2](#_Toc121332138)

[2. Введение………………………………………………………………..…....4](#_Toc121332139)

3. Графическое представление связей между таблицами ………………….5

4. Структуры таблиц баз данных…………................……………………......6

5. Содержание таблиц баз данных…………………………………………....9

6. Руководство пользователя………….…………………………………......11

7. Тексты запросов, примеры работ….……………………………………...16

8. Анализ результатов и выводы…….………………………………..……..12

9. Использованные источники….…………………………….……………...12

# **1. Задание**

Сформировать несколько таблиц.

Предусмотреть: ввод данных, редактирование, просмотр данных. Обязательные требования к базе данных: наличие таблиц-справочников и таблиц, использующих справочники; предусмотреть следующие роли: оператор базы данных; пользователь базы данных; администратор БД. Реализовать следующие типы запросов использованием нескольких таблиц:

* Для каждого вида детского товара указать сведения о нем (наименование, дату выпуска, поставщик, цена, для какого возраста предназначена, дата продажи и т.п.).
* Для каждого вида детского товара выдать список, отсортированный
* По дате выпуска,
* в алфавитном порядке по поставщику,
* по стоимости,
* по дате продажи,
* Найти самый дорогой детский товар, самый дешевый, среднюю стоимость по каждому виду и в целом.
* Найти детские товары с ценой в заданных пределах (предусмотреть ввод цены с клавиатуры) по каждому виду и в целом.
* Найти все детские товары заданного производителя (выбор производителя).
* Найти долю дешевых детских товаров (меньше заданного, ввод ограничения) от общего числа детских товаров (для каждого возраста (ввод) и в целом).
* Найти все детские товары с заданной датой выпуска (ввод даты).
* Найти все детские товары, чья дата продажи находится в заданных пределах (ввод интервала) для заданного производителя (выбор производителя) и в целом.
* Найти долю детских товаров, проданных за определенный период (ввод периода) от общего времени продажи.
* Найти долю дешевых игрушек (меньше заданного, ввод ограничения) от общего числа игрушек (для каждого возраста (ввод) и в целом).
* Найти все детские товары, поступившие от заданного поставщика (выбор поставщика), чья стоимость больше, чем средняя стоимость детских товаров, поступивших из заданной страны (выбор страны).
* Найти долю дешевых детских товаров (чья стоимость меньше заданной, ввод стоимости), поступивших от заданного поставщика (выбор поставщика) и в целом.
* Найти среднюю стоимость детских товаров, проданных за определенный промежуток времени (ввод интервала).
* Найти все детские товары, чья стоимость выше, чем средняя стоимость детских товаров заданного производителя (выбор производителя).
* Найти детские товары, которые лучше всего раскупают, указать параметры (стоимость, производитель, возраст и т.д.).

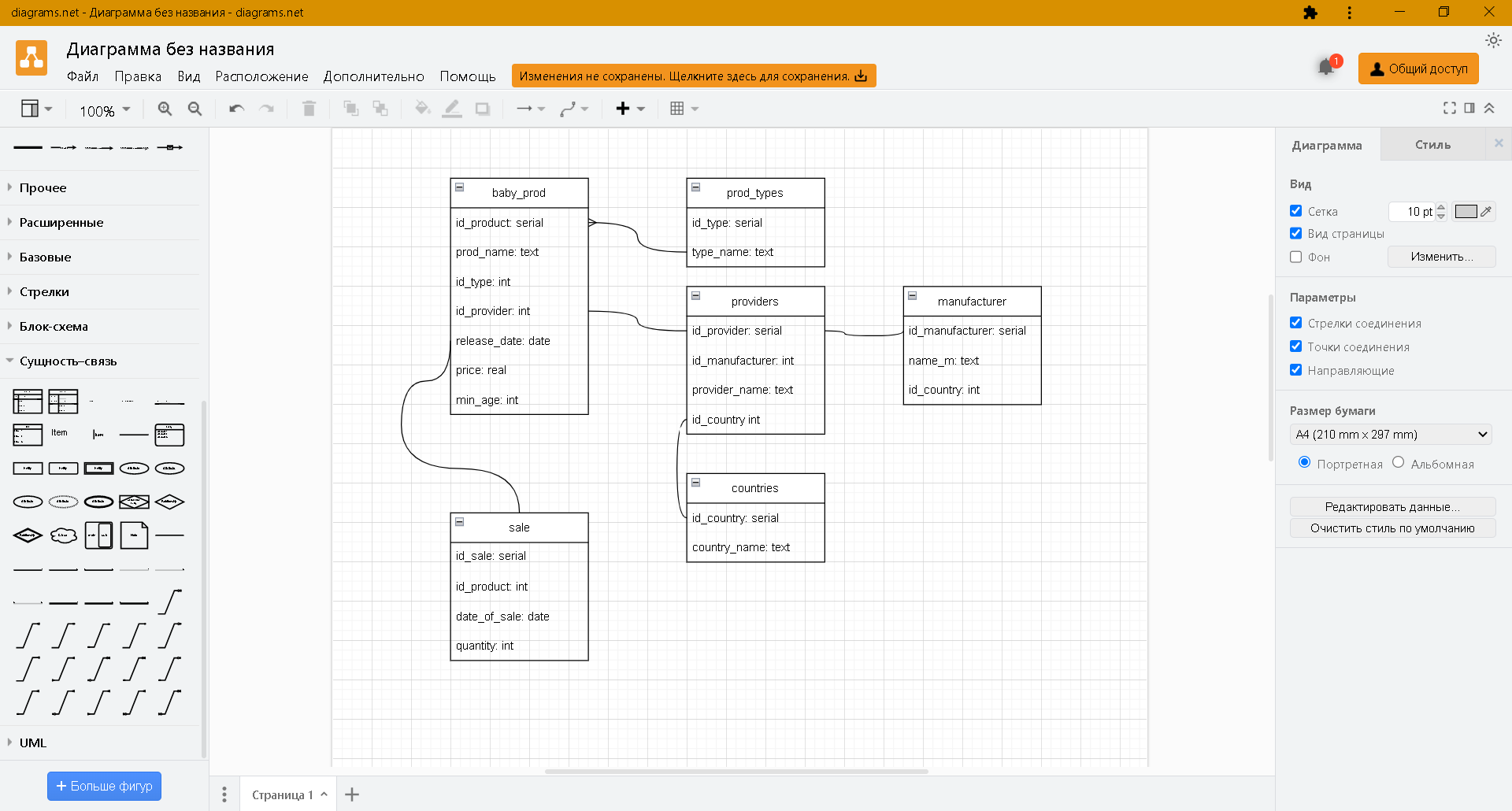
# **2. Введение**

База данных — это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).

На сегодняшний день любая современная организация не может обойтись без базы данных. Актуальность данной работы связана с существующей необходимостью в хранении, обработке больших объемов финансово-экономической информации. Наиболее эффективным способом решения этой проблемы является создание базы данных.

Целью работы является построение функционирующей базу данных, используя PostgreSQL, а также улучшение и закрепление практических навыков по работе с языком запросов SQL.

# **3. Графическое представление связей между таблицами**



*Рисунок 1 – Графическое представление связей между таблицами.*

**4. Структуры таблиц баз данных**

Таблица-справочник countries содержит информацию о странах (Таблица 1).

Таблица 1 – Структура countries

|  |  |  |  |
| --- | --- | --- | --- |
| countries | | | |
| Название поля | Тип данных | Ограничения | Комментарий |
| id\_country | serial | primary key, not null | идентификатор страны |
| country\_name | text | not null | название страны |

Таблица-справочник prod\_types содержит информацию о видах детских товаров (Таблица 2).

Таблица 2 – Структура prod\_types

|  |  |  |  |
| --- | --- | --- | --- |
| prod\_types | | | |
| Название поля | Тип данных | Ограничения | Комментарий |
| id\_type | serial | primary key, not null | идентификатор вида детских товаров |
| type\_name | text | - | название вида детского товара |

Таблица providers содержит информацию о поставщике (Таблица 3).

Таблица 3 – Структура providers

|  |  |  |  |
| --- | --- | --- | --- |
| providers | | | |
| Название поля | Тип данных | Ограничения | Комментарий |
| id\_provider | serial | primary key, not null | идентификатор поставщика |
| id\_manufact | int | foreign key | идентификатор производителя |
| provider\_name | text | not null | имя поставщика |
| id\_country | int | foreign key, not null | идентификатор страны поставщика |

Таблица manufacturers содержит информацию о производителе (Таблица 4). *Таблица 4 – Структура manufacturers*

|  |  |  |  |
| --- | --- | --- | --- |
| manufacturers | | | |
| Название поля | Тип данных | Ограничения | Комментарий |
| id\_manufact | serial | primary key, not null | идентификатор производителя |
| m\_name | text | not null | имя производителя |
| id\_country | int | foreign key, not null | идентификатор страны производителя |

Таблица baby\_prod содержит информацию о детских товарах (Таблица 5).

Таблица 5 – Структура baby\_prod

|  |  |  |  |
| --- | --- | --- | --- |
| baby\_prod | | | |
| Название поля | Тип данных | Ограничения | Комментарий |
| id\_product | serial | primary key, not null | идентификатор товара |
| prod\_name | text | - | имя товара |
| id\_provider | int | foreign key | идентификатор поставщика |
| id\_type | int | foreign key | идентификатор вида товара |
| release\_date | date | - | дата выпсука товара |
| price | numeric | - | текущая цена товара |
| min\_age | int | - | минимальный возраст предназначения товара |

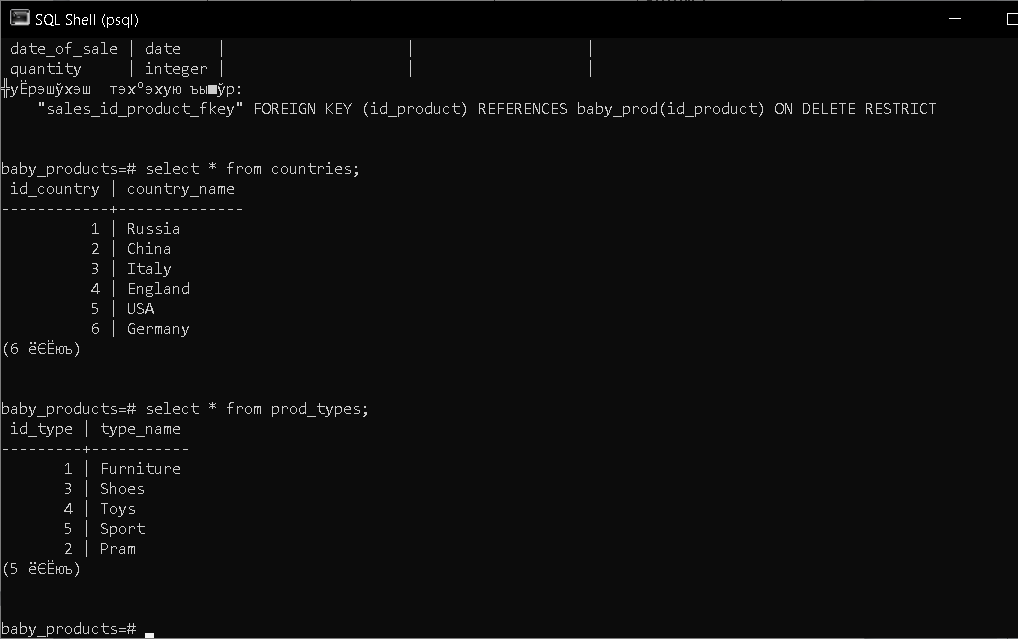
Таблица sales содержит информацию о продажах товара (Таблица 6).

Таблица 6 – Структура sales

|  |  |  |  |
| --- | --- | --- | --- |
| sales | | | |
| Название поля | Тип данных | Ограничения | Комментарий |
| id\_sale | serial | primary key, not null | идентификатор  операции по продаже |
| id\_product | int | - | идентификатор  товара, которого продали |
| date\_of\_sale | date | - | дата продажи |
| quantity | int | - | количество проданного товара |

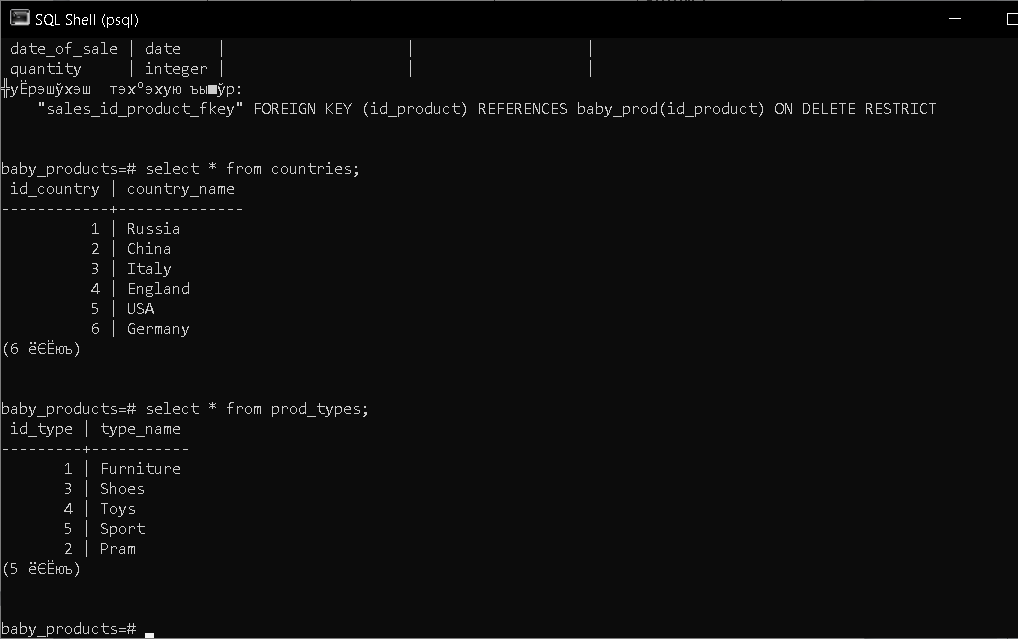
**5. Содержание таблиц баз данных**

Содержание таблицы-справочника countries:



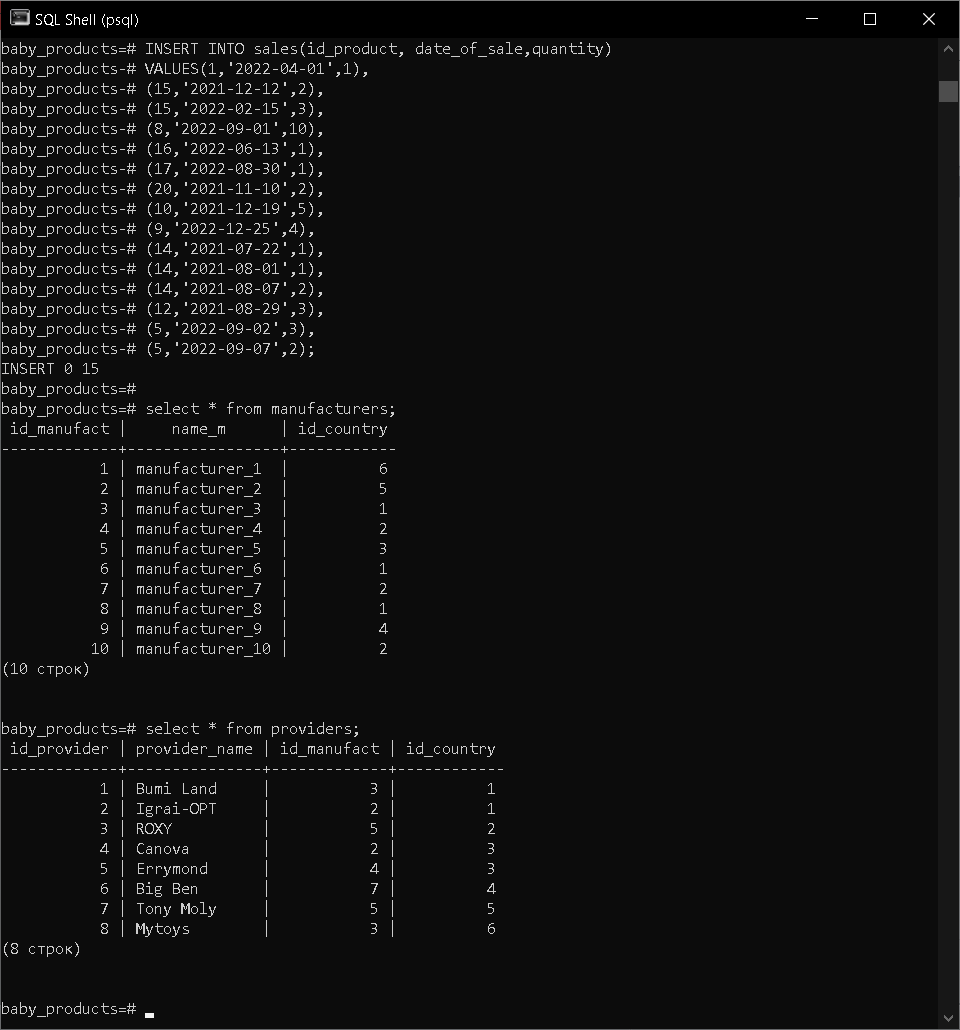
*Рисунок 2 – Содержание таблицы-справочника countries.*

Содержание таблицы-справочника product\_type:



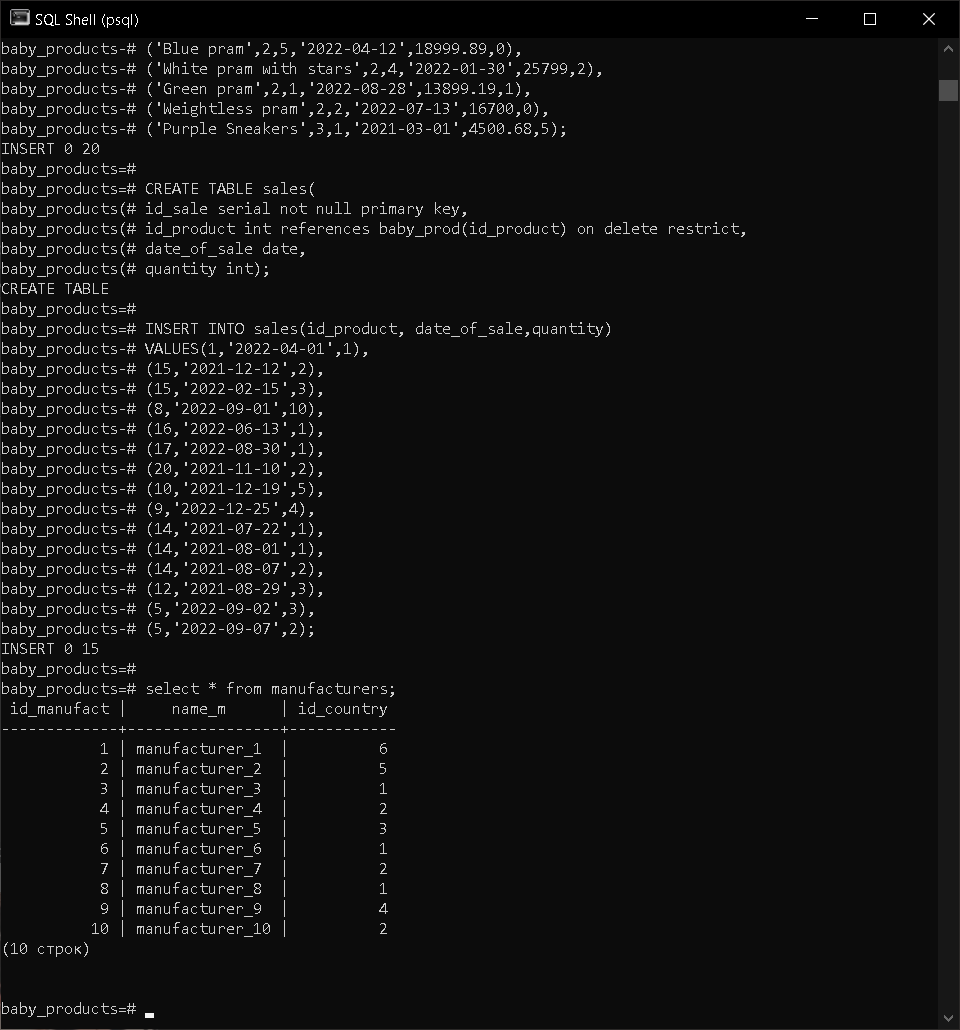
*Рисунок 3 – Содержание таблицы-справочника product\_type*

Содержание таблицы-справочника provider:



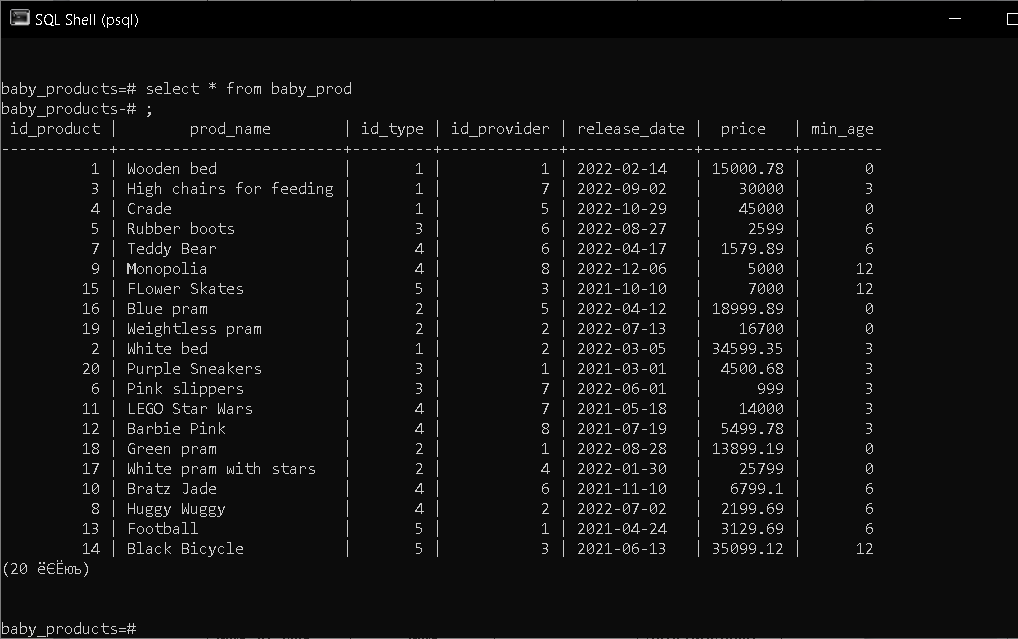
*Рисунок 4 – Содержание таблицы-справочника provider*

Содержание таблицы-справочника manufacturers:



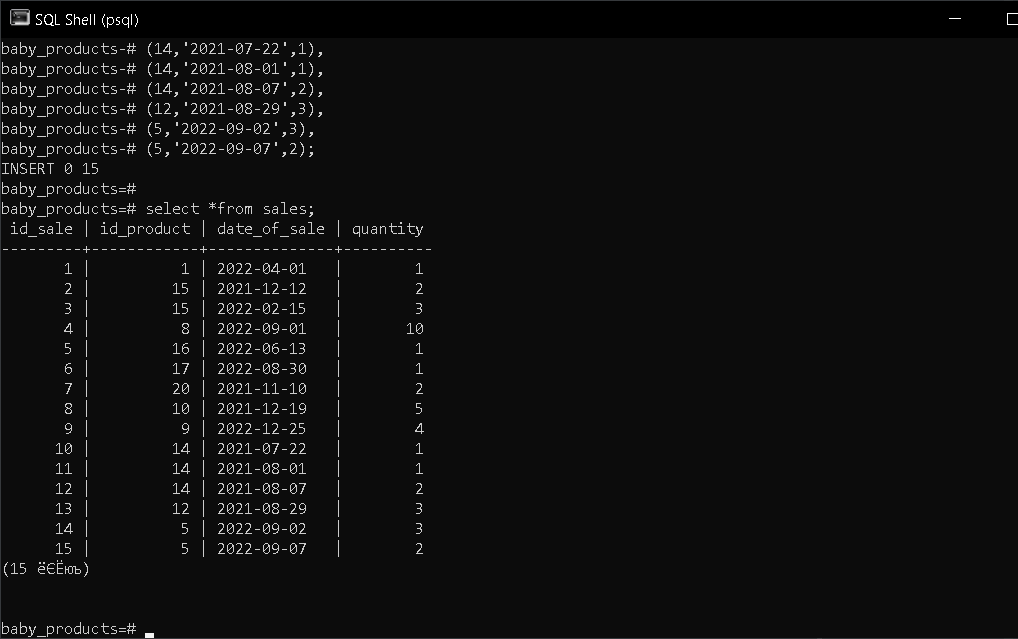
*Рисунок 5 – Содержание таблицы-справочника manufacturers*

Содержание таблицы baby\_prod:



*Рисунок 6– Содержание таблицы-справочника baby\_prod*

Содержание таблицы-справочника sales:



*Рисунок 7 – Содержание таблицы-справочника sales*

**6. Руководство пользователя**

*Таблица 8 – Описание функций*

|  |  |  |  |
| --- | --- | --- | --- |
| Номер задания, для которого написана функция | Наименование функции | Аргумент функции | Описание функции |
| 1 | baby\_prod\_release\_date\_info() | - | Выводит отсортированные сведения для каждого вида детского товара по дате выпуска |
| 2 | baby\_prod\_providers\_info() | - | Выводит отсортированные сведения для каждого вида детского товара в алфавитном порядке по поставщику |
| 3 | baby\_prod\_price\_info() | - | Выводит отсортированные сведения для каждого вида детского товара по стоимости |
| 4 | baby\_prod\_sale\_date\_info() | - | Выводит отсортированные сведения для каждого вида детского товара по дате продажи |
| 5 | the\_most\_expensive\_baby\_prod() | - | Возвращает самый дорогой детский товар по каждому виду |
| 6 | the\_cheapest\_baby\_prod() | - | Возвращает самый дешевый детский товар по каждому виду |
| 7 | avg\_prod\_price() | - | Возвращает среднюю цену продажи детских товаров в целом |
| 8 | avg\_types\_price() | - | Возвращает среднюю цену продажи по каждому виду |
| 9 | price\_between(a real, b real) | а int – нижний предел цены  b int – верхний предел цены | Возвращает детские товары с ценой в заданных пределах |
| 10 | manufacturs\_products(man text) | man text– наименование производителя | Возвращает все детские товары заданного производителя по наименованию производителя |
| 11 | group\_cheap\_products(pri real) | pri real – верхний предел цены | Возвращает долю дешевых детских товаров от общего числа детских товаров для каждого возраста |
| 12 | group\_cheap\_products(age int ,pri real) | pri real – верхний предел цены  age int- возраст | Возвращает долю дешевых детских товаров от общего числа детских товаров для введенного возраста |
| 13 | all\_cheap\_products(pri real) | pri real – верхний предел цены | Возвращает долю дешевых детских товаров от общего числа детских товаров в целом |
| 14 | product\_by\_release\_day(a date) | a date – дата выпуска товара | Возвращает все детские товары с заданной датой выпуска |
| 15 | product\_between\_sale\_days(a date, b date, man text) | a date –нижний предел даты продажи  b date - верхний предел даты продажи  man text – наименование производителя | Возвращает все детские товары, чья дата продажи находится в заданных пределах для заданного производителя |
| 16 | product\_between\_sale\_days(a date, b date) | a date –нижний предел даты продажи  b date - верхний предел даты продажи | Возвращает все детские товары, чья дата продажи находится в заданных пределах |
| 17 | part\_of\_products\_sales\_days(a date, b date) | a date –нижний предел даты продажи  b date - верхний предел даты продажи | Возвращает долю детских товаров, проданных за определенный период от общего времени продажи. |
| 18 | part\_of\_cheap\_toys(pri real) | pri real – верхний предел цены | Возвращает долю дешевых игрушек от общего числа игрушек для каждого возраста |
| 19 | part\_of\_cheap\_toys(age int,pri real) | pri real – верхний предел цены  age int- возраст | Возвращает долю дешевых игрушек от общего числа игрушек для заданного возраста |
| 20 | all\_part\_of\_cheap\_toys(pri real) | pri real – верхний предел цены | Возвращает долю дешевых игрушек от общего числа игрушек |
| 21 | provider\_of\_product\_by\_name(pr\_name text, country text) | pr\_name text – имя поставщика  country text - название страны | Возвращает все детские товары, поступившие от заданного поставщика, чья стоимость больше, чем средняя стоимость детских товаров, поступивших из заданной страны |
| 22 | providers\_cheap\_products(pri real, provider text) | pri real – верхний предел цены  provider text – имя поставщика | Возвращает долю дешевых детских товаров, поступивших от заданного поставщика |
| 23 | avg\_price\_in\_period(a date, b date) | a date –нижний предел даты продажи  b date - верхний предел даты продажи | Возвращает среднюю стоимость детских товаров, проданных за определенный промежуток времени |
| 24 | price\_of\_product\_more\_than\_avg(man text) | man text- название производителя | Возвращает все детские товары, чья стоимость выше, чем средняя стоимость детских товаров заданного производителя |
| 25 | popular\_baby\_prod(pri real , man text, age int) | pri real –цена  man text- название производителя  age int- возраст | Возвращает детские товары, которые лучше всего раскупают, указать параметры (стоимость, производитель, возраст) |
| 26 | popular\_baby\_prod(man text) | man text- название производителя | Возвращает детские товары, которые лучше всего раскупают у определенного производителя |
| 27 | popular\_baby\_prod(age int) | age int- возраст | Возвращает детские товары определенной возрастной категории, которые лучше всего раскупают |
| 28 | popular\_baby\_prod(pri real) | pri real –цена | Возвращает детские товары, которые лучше всего раскупают по заданной цене |

**7. Тексты запросов, примеры работ**

1. Создадим следующие роли: оператор БД, пользователь БД, администратор БД.

**Оператор БД (**пополнение справочников).

**Текст запроса:**

CREATE ROLE operator\_role WITH password '123' LOGIN;

GRANT INSERT, UPDATE, SELECT ON countries, providers, sales, prod\_types, manufacturers TO operator\_role;

GRANT USAGE, SELECT ON sales\_id\_sale\_seq,countries\_id\_country\_seq,providers\_id\_provider\_seq ,manufacturers\_id\_manufact\_seq, prod\_types\_id\_type\_seq TO operator\_role;

**Пользователь БД** (основная работа с БД, с ограничениями для некоторого вида операций).

**Текст запроса:**

CREATE ROLE user\_role WITH password '456' LOGIN;

GRANT SELECT ON ALL TABLES IN SCHEMA public TO user\_role;

GRANT INSERT ON ALL TABLES IN SCHEMA public TO user\_role;

GRANT UPDATE ON ALL TABLES IN SCHEMA public TO user\_role;

REVOKE DELETE ON countries, providers, manufacturers, sales FROM user\_role;

GRANT USAGE, SELECT ON sales\_id\_sale\_seq,countries\_id\_country\_seq,providers\_id\_provider\_seq ,manufacturers\_id\_manufact\_seq, prod\_types\_id\_type\_seq, baby\_prod\_id\_product\_seq TO operator\_role;

**Администратор БД** (любые изменения БД).

**Текст запроса:**

CREATE ROLE admin\_role WITH CREATEDB CREATEROLE password '0000' LOGIN;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin\_role;

Проверим как работают роли. Например, оператор может пополнять (INSERT, UPDATE, т.к. для UPDATE нужны права на SELECT, добавим права и на эту операцию) только справочники.

**Текст запроса:**

\c baby\_products operator\_role;

INSERT INTO sales(id\_product, date\_of\_sale,quantity)

VALUES(1,'2022-07-01',2);

SELECT \*from sales where id\_sale=16;

DELETE from sales where id\_sale=16;

UPDATE providers set id\_country=5 where id\_provider=5;

\c baby\_products user\_role;

INSERT INTO sales(id\_product, date\_of\_sale,quantity)

VALUES(4,'2022-07-01',2);

SELECT \*from baby\_prod where id\_product=4;

UPDATE providers set id\_country=5 where id\_provider=5;

DELETE from sales where id\_sale=16;

\c baby\_products admin\_role;

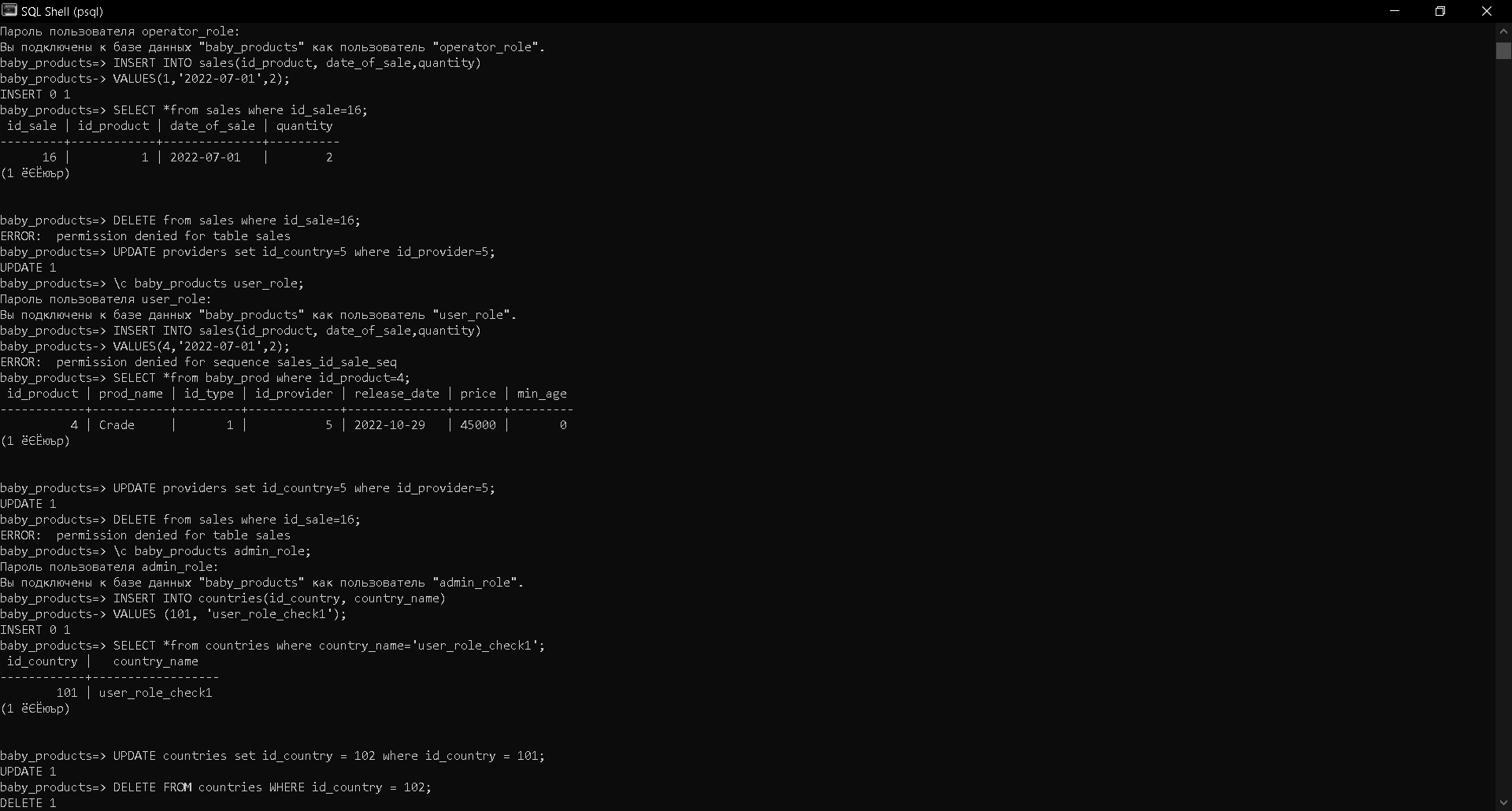
INSERT INTO countries(id\_country, country\_name)

VALUES (101, 'user\_role\_check1');

SELECT \*from countries where country\_name='user\_role\_check1';

UPDATE countries set id\_country = 102 where id\_country = 101;

DELETE FROM countries WHERE id\_country = 102;



*Рисунок 8 – Проверка работы ролей.*

1. Для каждого вида детского товара выдать список, отсортированный

**• по дате выпуска**

**Текст запроса:**

create or replace function baby\_prod\_release\_date\_info()

returns table (release\_date date, product\_name text, type\_name text, provider\_name text,producing\_country text, price real)as $$

begin

return query

select baby\_prod.release\_date, baby\_prod.prod\_name as product\_name, prod\_types.type\_name, providers.provider\_name, countries.country\_name as producing\_country , baby\_prod.price

from baby\_prod

join prod\_types on baby\_prod.id\_type = prod\_types.id\_type

join providers on baby\_prod.id\_provider=providers.id\_provider

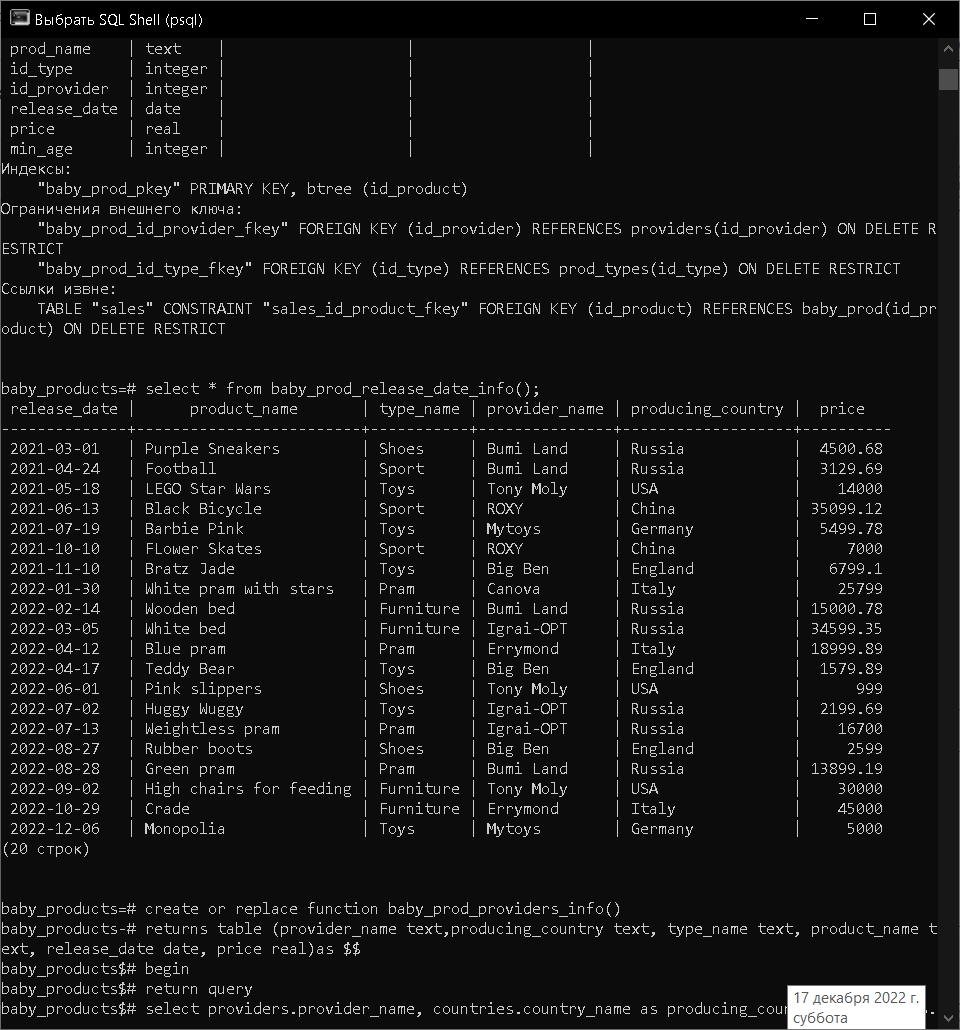
join countries on providers.id\_country=countries.id\_country

order by baby\_prod.release\_date;

end $$

language plpgsql;

select \* from baby\_prod\_release\_date\_info();



*Рисунок 9 – Результат запроса-* *список, отсортированный по дате выпуска*

**• в алфавитном порядке по поставщику**

**Текст запроса:**

create or replace function baby\_prod\_providers\_info()

returns table (provider\_name text,producing\_country text, type\_name text, product\_name text, release\_date date, price real)as $$

begin

return query

select providers.provider\_name, countries.country\_name as producing\_country, prod\_types.type\_name, baby\_prod.prod\_name as product\_name, baby\_prod.release\_date, baby\_prod.price

from baby\_prod

join prod\_types on baby\_prod.id\_type = prod\_types.id\_type

join providers on baby\_prod.id\_provider=providers.id\_provider

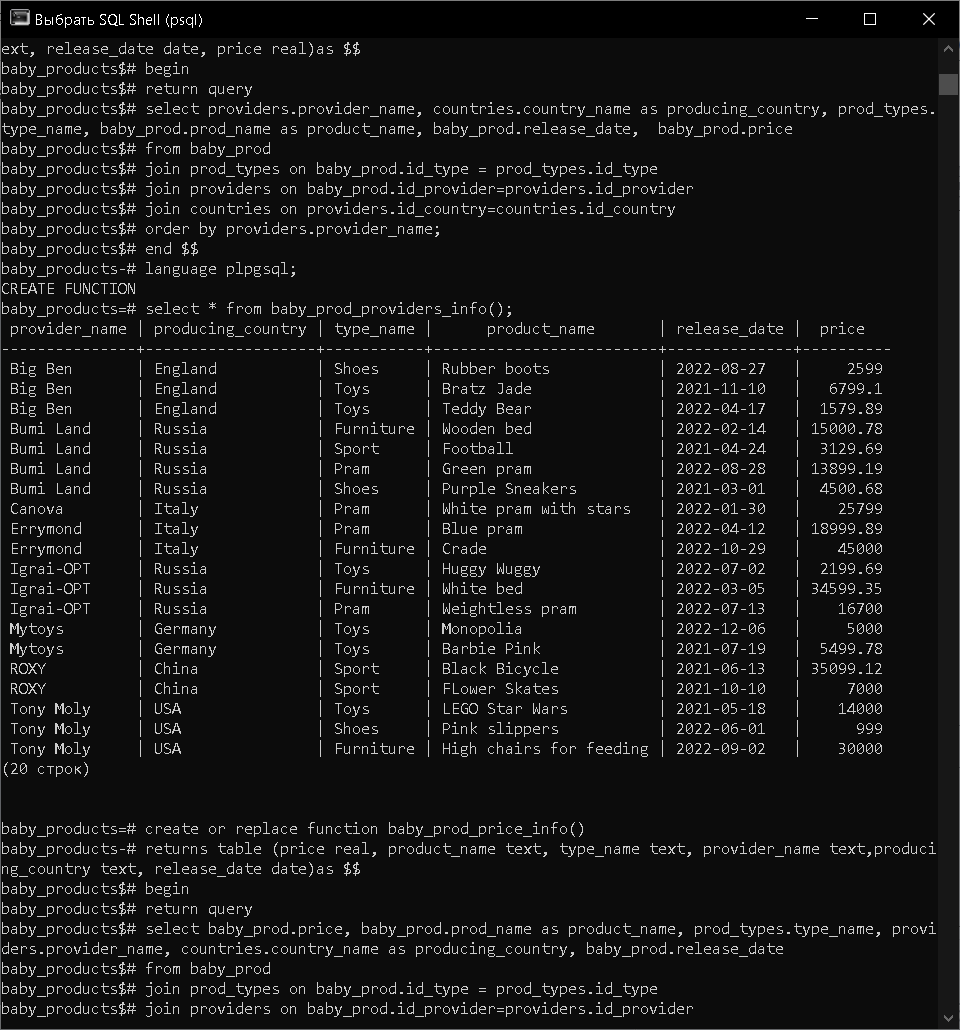
join countries on providers.id\_country=countries.id\_country

order by providers.provider\_name;

end $$

language plpgsql;

select \* from baby\_prod\_providers\_info();



*Рисунок 10 – Результат запроса-* *список, отсортированный в алфавитном порядке по поставщику*

**• по стоимости**

**Текст запроса:**

create or replace function baby\_prod\_price\_info()

returns table (price real, product\_name text, type\_name text, provider\_name text,producing\_country text, release\_date date)as $$

begin

return query

select baby\_prod.price, baby\_prod.prod\_name as product\_name, prod\_types.type\_name, providers.provider\_name, countries.country\_name as producing\_country, baby\_prod.release\_date

from baby\_prod

join prod\_types on baby\_prod.id\_type = prod\_types.id\_type

join providers on baby\_prod.id\_provider=providers.id\_provider

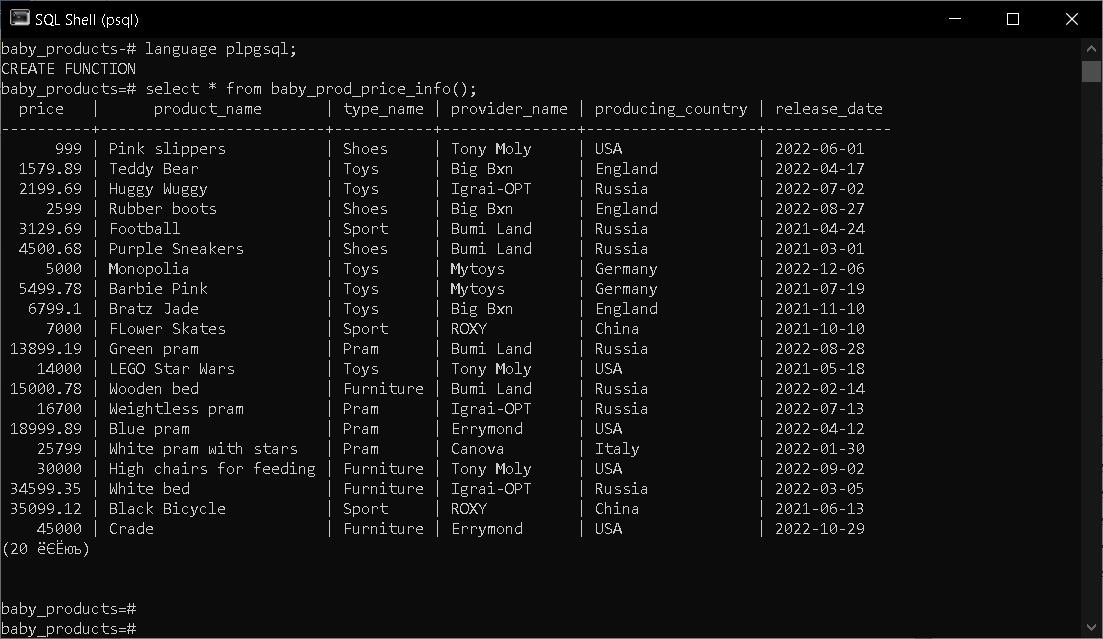
join countries on providers.id\_country=countries.id\_country

order by baby\_prod.price;

end $$

language plpgsql;

select \* from baby\_prod\_price\_info();



*Рисунок 11 – Результат запроса-* *список, отсортированный по стоимости*

**• по дате продажи**

**Текст запроса:**

create or replace function baby\_prod\_sale\_date\_info()

returns table (date\_of\_sale date, product\_name text, type\_name text, price real, provider\_name text, producing\_country text, release\_date date)as $$

begin

return query

select sales.date\_of\_sale, baby\_prod.prod\_name as product\_name, prod\_types.type\_name,baby\_prod.price, providers.provider\_name, countries.country\_name as producing\_country, baby\_prod.release\_date

from baby\_prod

join sales on baby\_prod.id\_product=sales.id\_product

join prod\_types on baby\_prod.id\_type = prod\_types.id\_type

join providers on baby\_prod.id\_provider=providers.id\_provider

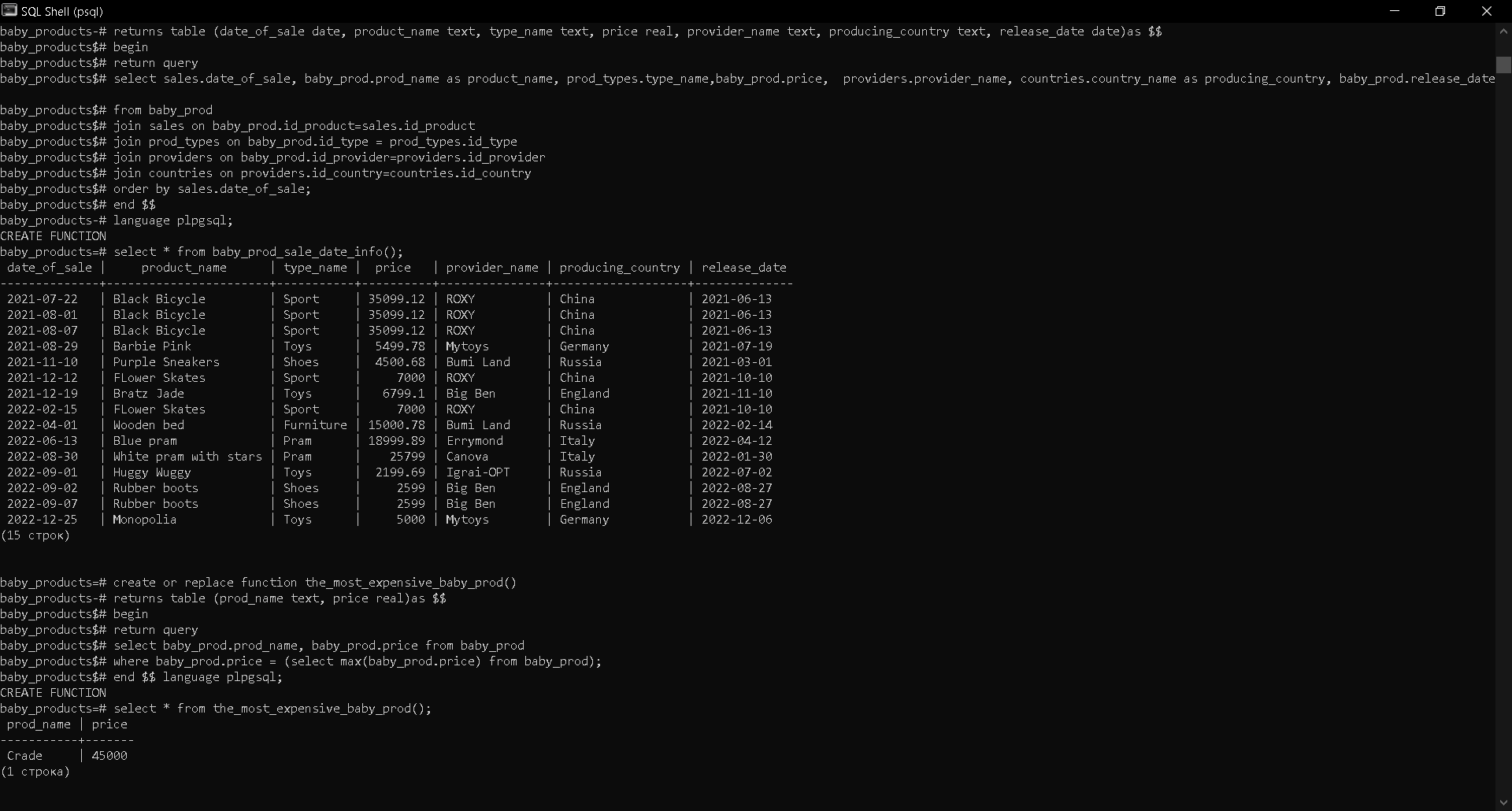
join countries on providers.id\_country=countries.id\_country

order by sales.date\_of\_sale;

end $$

language plpgsql;

select \* from baby\_prod\_sale\_date\_info();



*Рисунок 12 – Результат запроса -* *список, отсортированный по дате продажи*

1. Найти самый дорогой детский товар, самый дешевый, среднюю стоимость по каждому виду и в целом.

**• самый дорогой товар**

**Текст запроса:**

create or replace function the\_most\_expensive\_baby\_prod()

returns table (prod\_name text, price real)as $$

begin

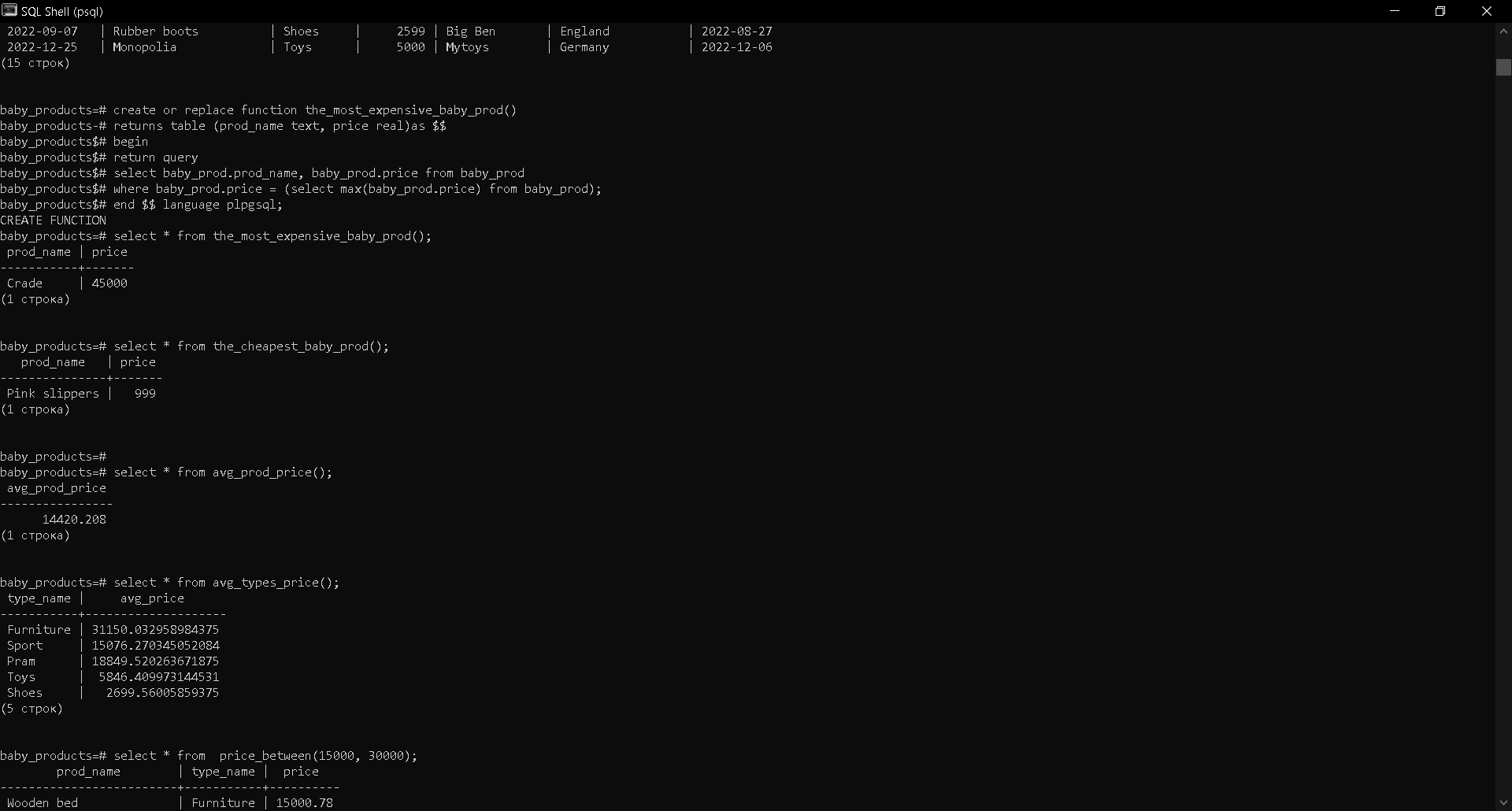
return query

select baby\_prod.prod\_name, baby\_prod.price from baby\_prod

where baby\_prod.price = (select max(baby\_prod.price) from baby\_prod);

end $$ language plpgsql;

select \* from the\_most\_expensive\_baby\_prod();



*Рисунок 13 – Результат запроса -* *самый дорогой товар.*

**• самый дешевый**

**Текст запроса:**

create or replace function the\_cheapest\_baby\_prod()

returns table (prod\_name text, price real)as $$

begin

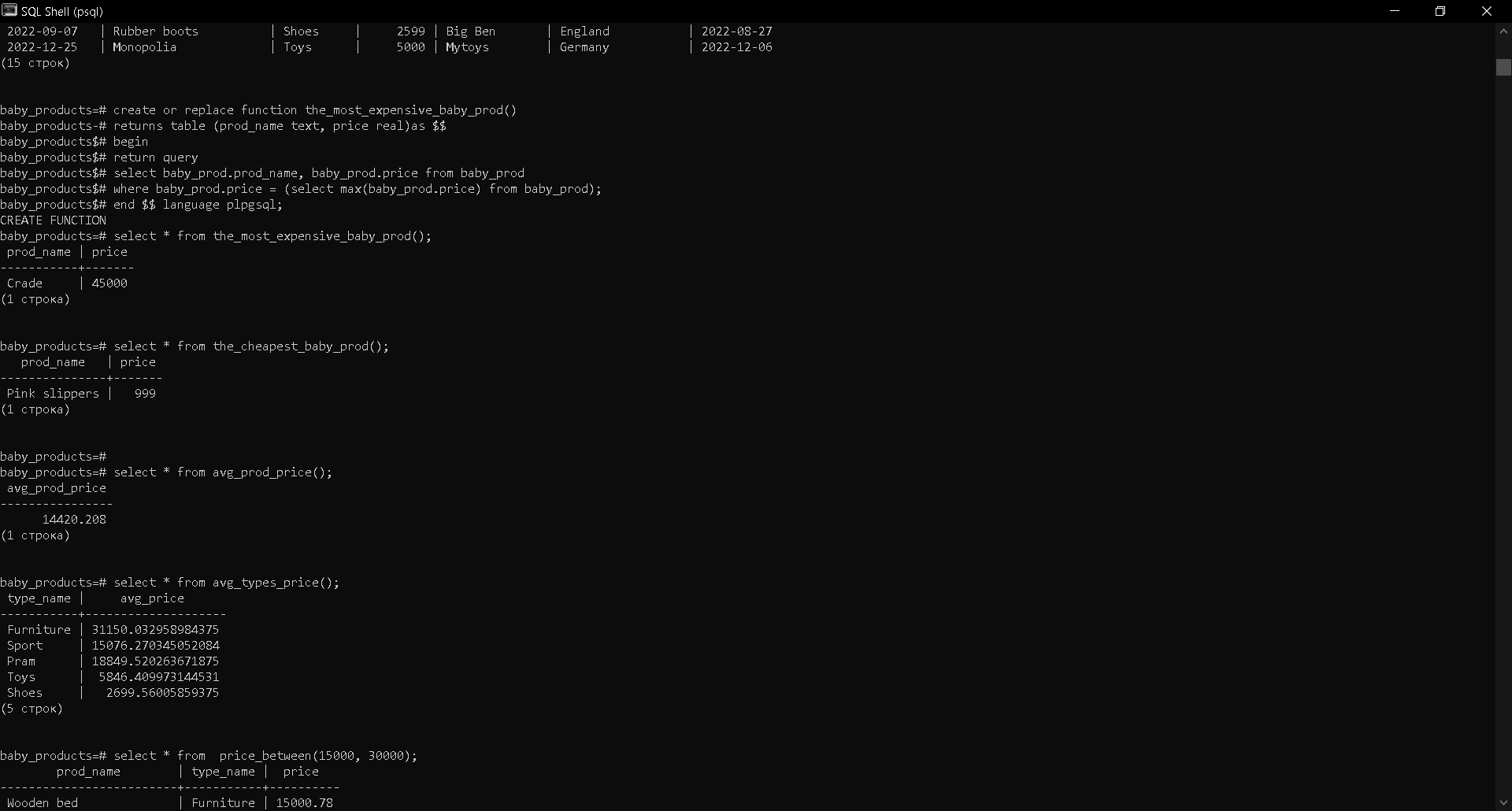
return query

select baby\_prod.prod\_name, baby\_prod.price from baby\_prod

where baby\_prod.price = (select min(baby\_prod.price) from baby\_prod);

end $$ language plpgsql;

select \* from the\_cheapest\_baby\_prod();



*Рисунок 14 – Результат запроса -* *самый дешевый товар.*

**• средняя стоимость**

**Текст запроса:**

create or replace function avg\_prod\_price()

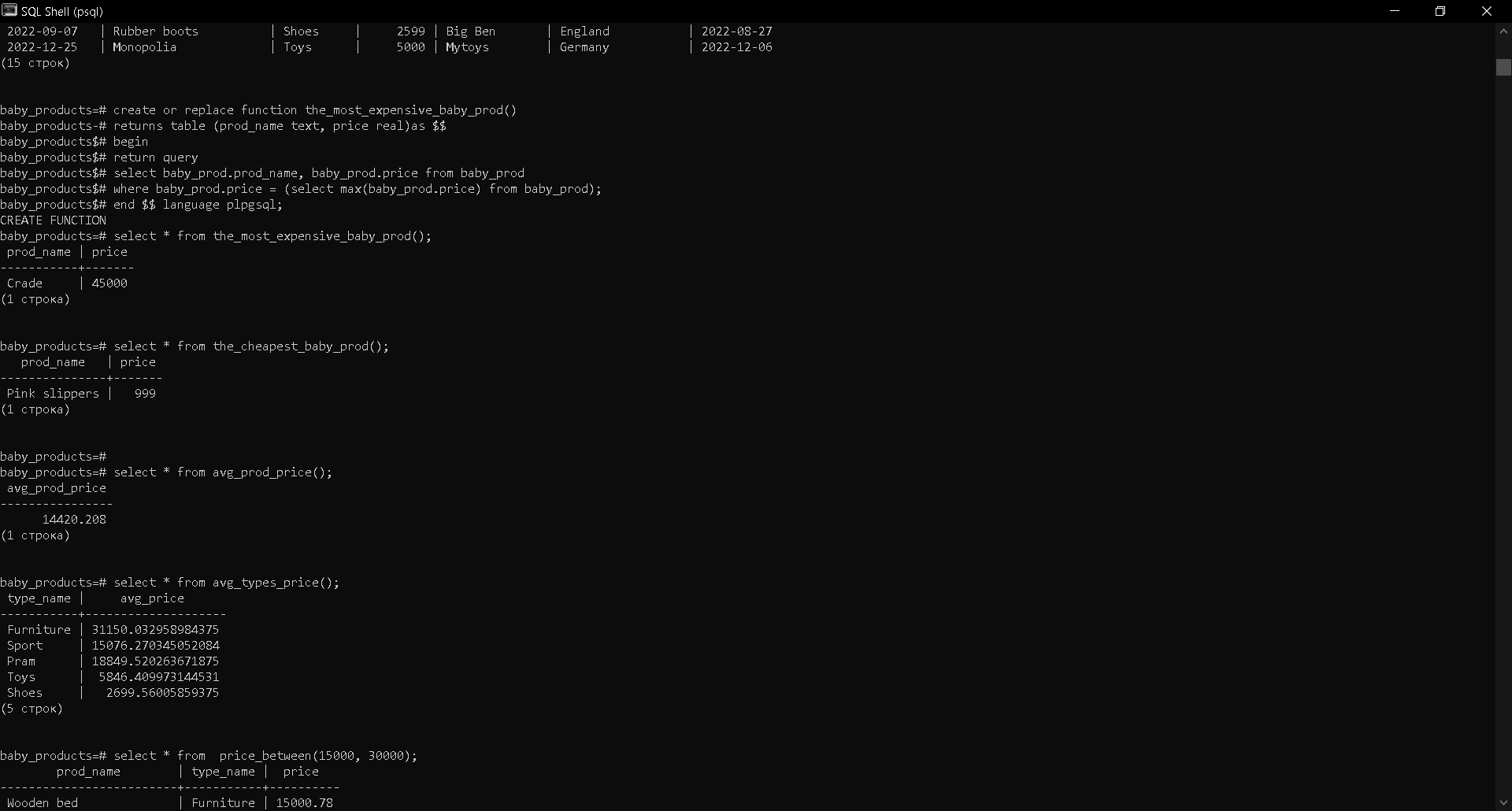
returns real as $$

begin

return (select avg(baby\_prod.price) from baby\_prod);

end $$ language plpgsql;

select \* from avg\_prod\_price();



*Рисунок 15 – Результат запроса –* *средняя стоимость.*

**• средняя стоимость каждого вида**

**Текст запроса:**

create or replace function avg\_types\_price()

returns table (type\_name text, avg\_price float)as $$

begin

return query

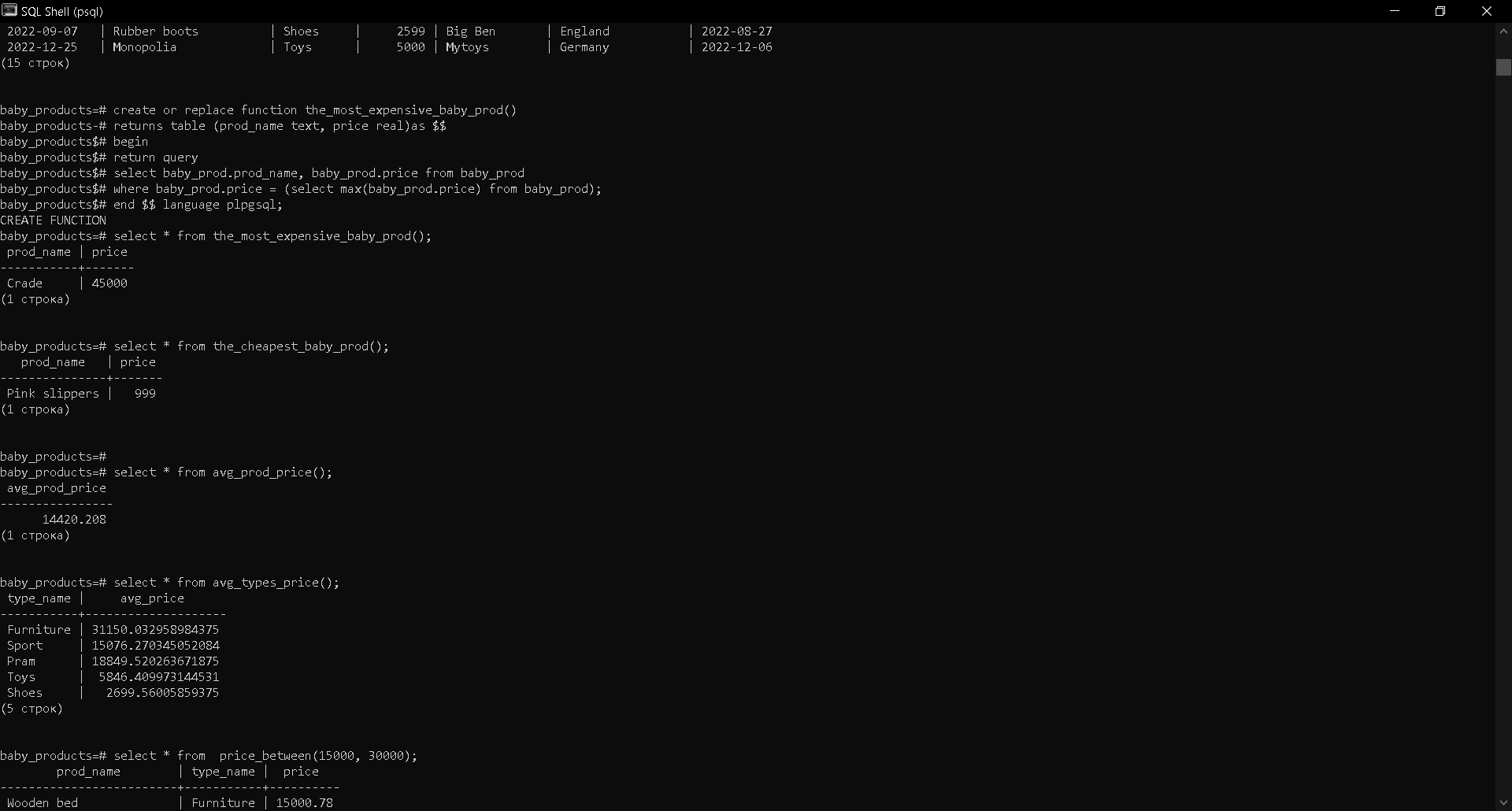
select prod\_types.type\_name as type\_name, avg(baby\_prod.price)

from baby\_prod join prod\_types on baby\_prod.id\_type = prod\_types.id\_type

group by prod\_types.type\_name;

end $$ language plpgsql;

select \* from avg\_types\_price();



*Рисунок 16 – Результат запроса –* *средняя стоимость.*

1. Найти детские товары с ценой в заданных пределах (предусмотреть ввод цены с клавиатуры) по каждому виду и в целом.

**Текст запроса:**

create function price\_between(a real, b real)

returns table (prod\_name text, type\_name text, price real) as $$

BEGIN

return query

select baby\_prod.prod\_name , prod\_types.type\_name, baby\_prod.price from baby\_prod

join prod\_types on baby\_prod.id\_type=prod\_types.id\_type

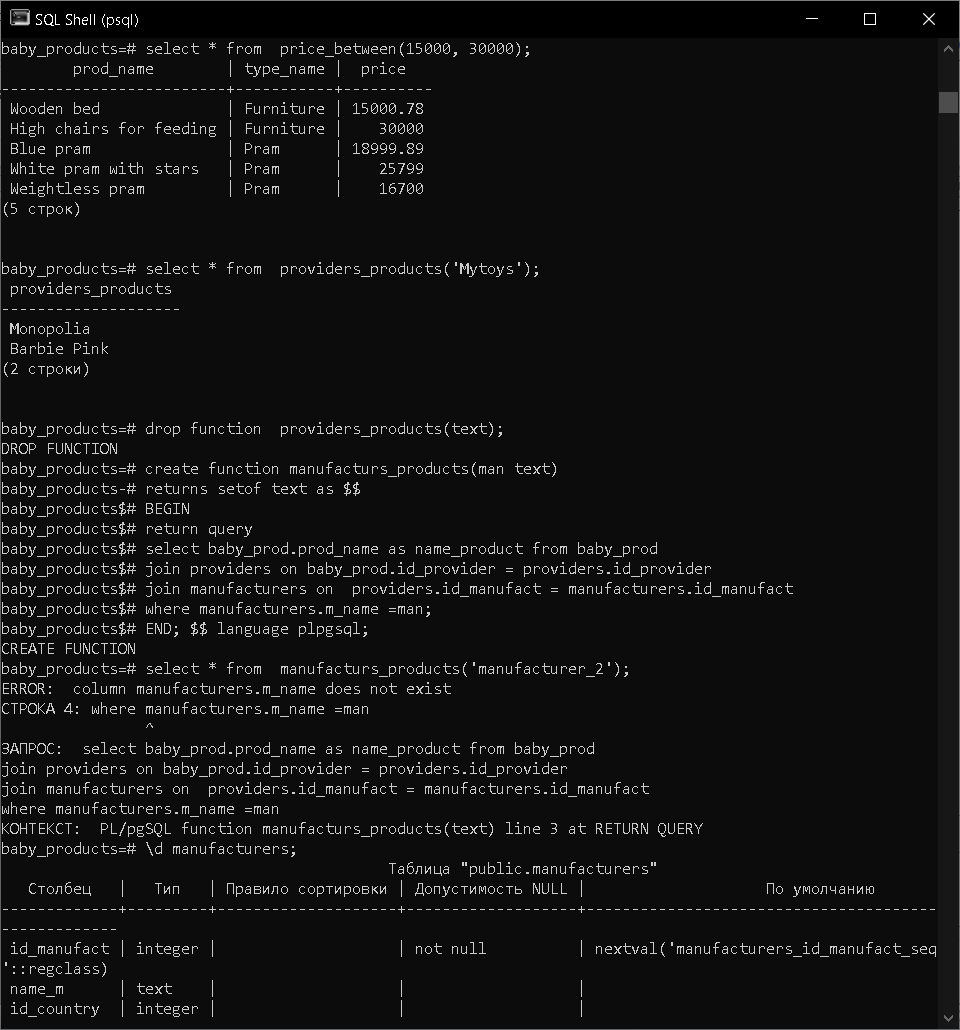
where baby\_prod.price::numeric::real >= a

and baby\_prod.price::numeric::real <= b;

END;

$$ language plpgsql;

select \* from price\_between(15000, 30000);



*Рисунок 17 – Результат запроса.*

1. Найти все детские товары заданного производителя (выбор производителя).

**Текст запроса:**

create or replace function manufacturs\_products(man text)

returns table ( id\_ int, manufacturs\_products text, manufact\_name text)

as $$

BEGIN

return query select baby\_prod.id\_product, baby\_prod.prod\_name, manufacturers.name\_m from baby\_prod

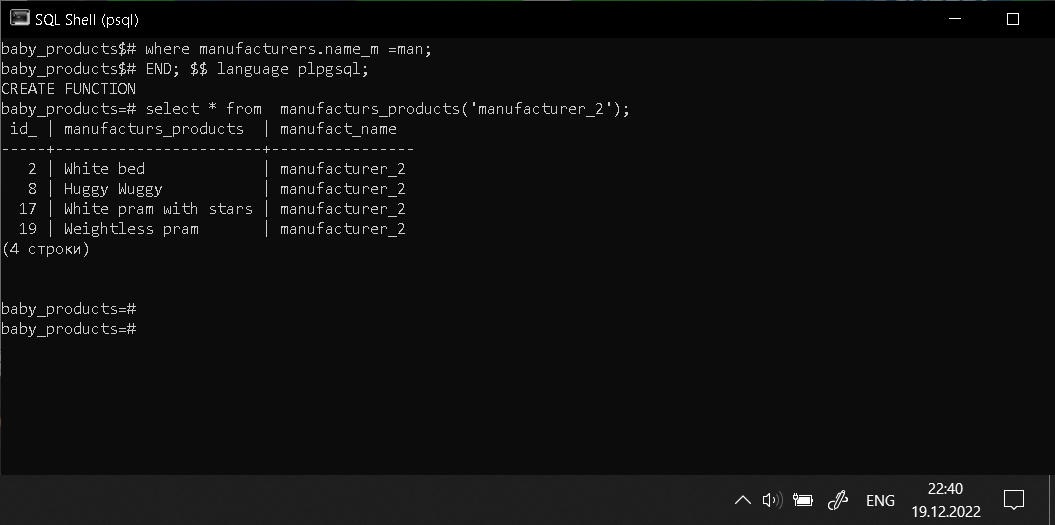
join providers on baby\_prod.id\_provider = providers.id\_provider

join manufacturers on providers.id\_manufact = manufacturers.id\_manufact

where manufacturers.name\_m =man;

END; $$ language plpgsql;

select \* from manufacturs\_products('manufacturer\_2');



*Рисунок 18 – Результат запроса.*

1. Найти долю дешевых детских товаров (меньше заданного, ввод ограничения) от общего числа детских товаров (для каждого возраста (ввод) и в целом).

* Для поиска доли детских товаров от общего числа детских товаров для каждого возраста сделаем перегруженную функцию. При вводе возраста функция будет выводить долю товаров только для определённого возраста.

**Текст запроса:**

create or replace function group\_cheap\_products(pri real)

returns table ( min\_age int, count bigint, part float, count\_of\_products bigint)

as $$

declare all\_count bigint;

BEGIN

all\_count = (select count(\*) from baby\_prod);

return query (select baby\_prod.min\_age, count (id\_product) as count, count(id\_product)/all\_count::bigint::float as part, all\_count

from baby\_prod where baby\_prod.price::numeric::int <= pri

group by baby\_prod.min\_age

order by baby\_prod.min\_age);

END; $$ language plpgsql;

select \* from group\_cheap\_products(10000);

create or replace function group\_cheap\_products(age int ,pri real)

returns table ( min\_age int, count bigint, part float, all\_count bigint)

as $$

declare all\_count bigint;

BEGIN

all\_count = (select count(\*) from baby\_prod);

return query (select baby\_prod.min\_age, count (id\_product) as count, count(id\_product)/all\_count::bigint::float as part, all\_count

from baby\_prod

where baby\_prod.price::numeric::int <= pri

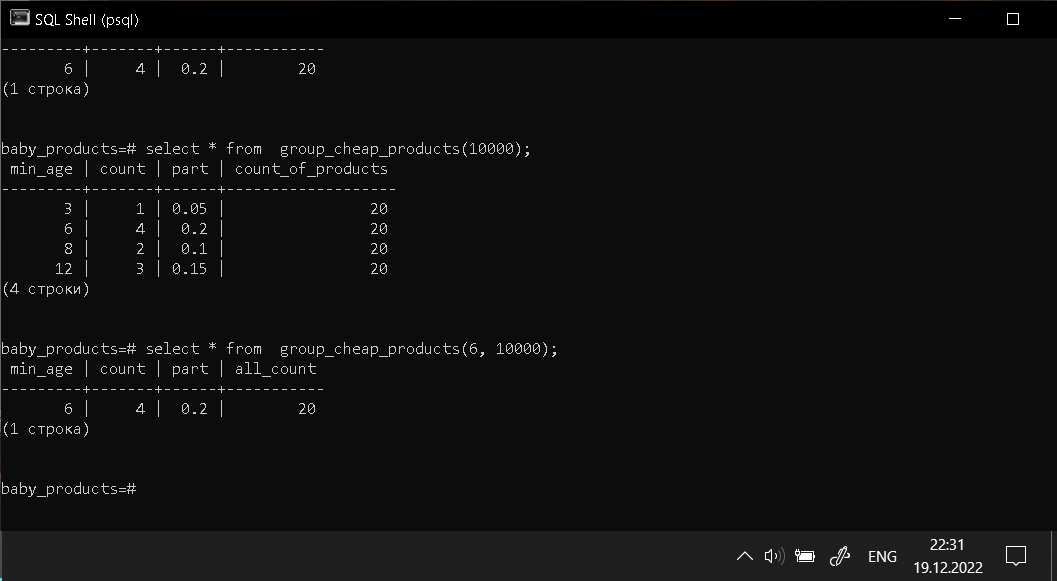
and baby\_prod.min\_age=age

group by baby\_prod.min\_age

order by baby\_prod.min\_age);

END; $$ language plpgsql;

select \* from group\_cheap\_products(6, 10000);



*Рисунок 19 – Результат запроса.*

* Поиск доли дешевых детских товаров от общего числа детских товаров в целом.

**Текст запроса:**

create or replace function all\_cheap\_products(pri real)

returns table (count\_of\_cheap bigint, part float, count\_of\_products int) as $$

declare all\_count int;

BEGIN

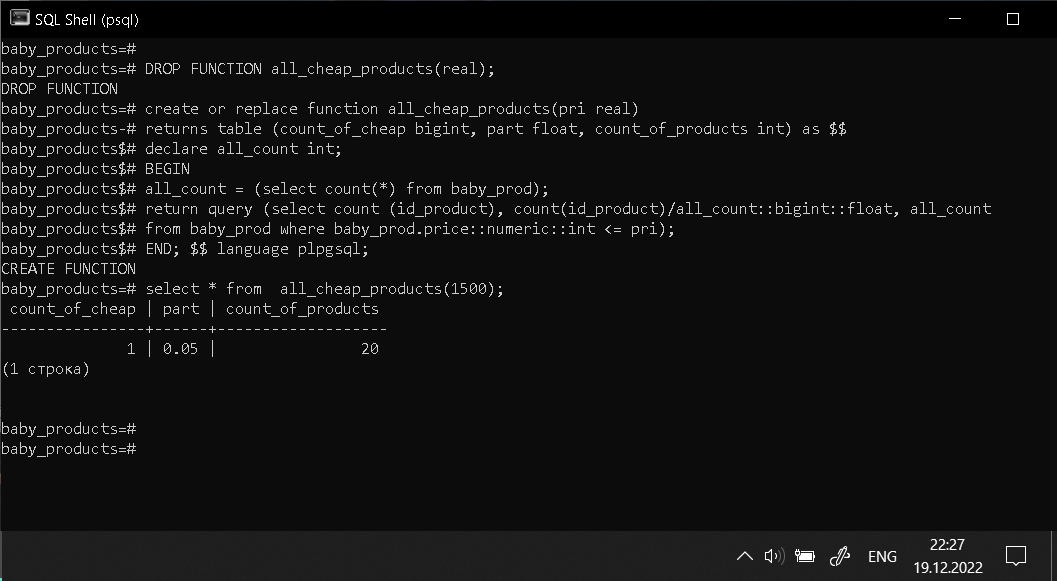
all\_count = (select count(\*) from baby\_prod);

return query (select count (id\_product), count(id\_product)/all\_count::bigint::float, all\_count

from baby\_prod where baby\_prod.price::numeric::int <= pri);

END; $$ language plpgsql;

select \* from all\_cheap\_products(1500);



*Рисунок 20 – Результат запроса.*

1. Найти все детские товары с заданной датой выпуска (ввод даты).

**Текст запроса:**

create or replace function product\_by\_release\_day(a date)

returns table(id\_product int, prod\_name text, r\_date date)

as $$

begin

return query select baby\_prod.id\_product, baby\_prod.prod\_name ,

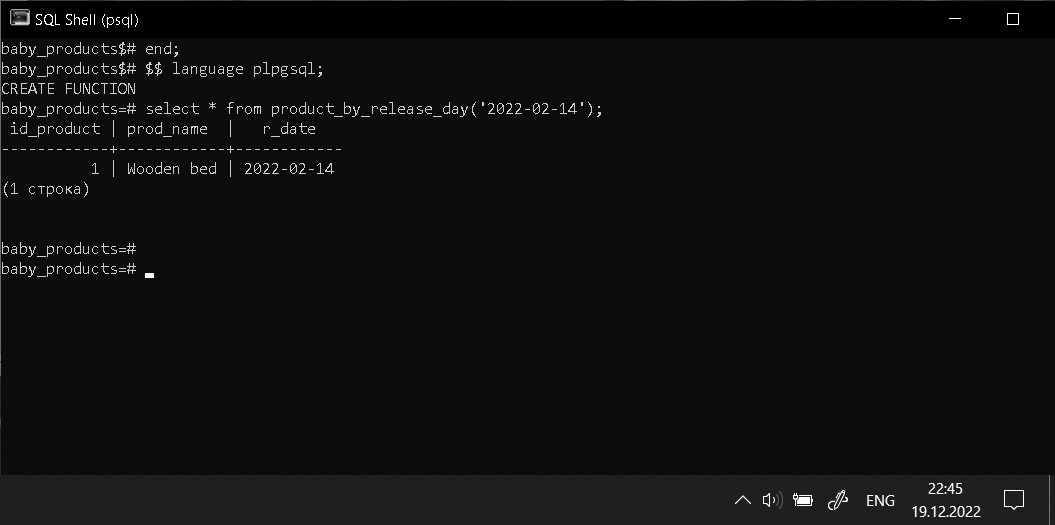
baby\_prod.release\_date from baby\_prod

where baby\_prod.release\_date=a;

end;

$$ language plpgsql;

select \* from product\_by\_release\_day('2022-02-14');



*Рисунок 21 – Результат запроса.*

1. Найти все детские товары, чья дата продажи находится в заданных пределах (ввод интервала) для заданного производителя (выбор производителя) и в целом.

* Для заданного запроса создадим перегруженную функцию, которая при вводе имени производителя выводит список для заданного производителя, иначе выводит список в целом.

**Текст запроса:**

create or replace function product\_between\_sale\_days(a date, b date, man text)

returns table(id\_product int,prod\_name text, manufacturer text, r\_date date)

as $$

begin

return query select baby\_prod.id\_product, baby\_prod.prod\_name ,manufacturers.name\_m,

sales.date\_of\_sale from baby\_prod

join sales on baby\_prod.id\_product=sales.id\_product

join providers on baby\_prod.id\_provider=providers.id\_provider

join manufacturers on providers.id\_manufact=manufacturers.id\_manufact

where

manufacturers.name\_m=man

and sales.date\_of\_sale>=a

and sales.date\_of\_sale<=b

order by sales.date\_of\_sale;

end;

$$ language plpgsql;

select \* from product\_between\_sale\_days('2021-02-14','2022-12-31','manufacturer\_3');

create or replace function product\_between\_sale\_days(a date, b date)

returns table(id\_product int, prod\_name text, provider text, r\_date date)

as $$

begin

return query select baby\_prod.id\_product, baby\_prod.prod\_name ,providers.provider\_name,

sales.date\_of\_sale from baby\_prod

join sales on baby\_prod.id\_product=sales.id\_product

join providers on baby\_prod.id\_provider=providers.id\_provider

where

sales.date\_of\_sale>=a

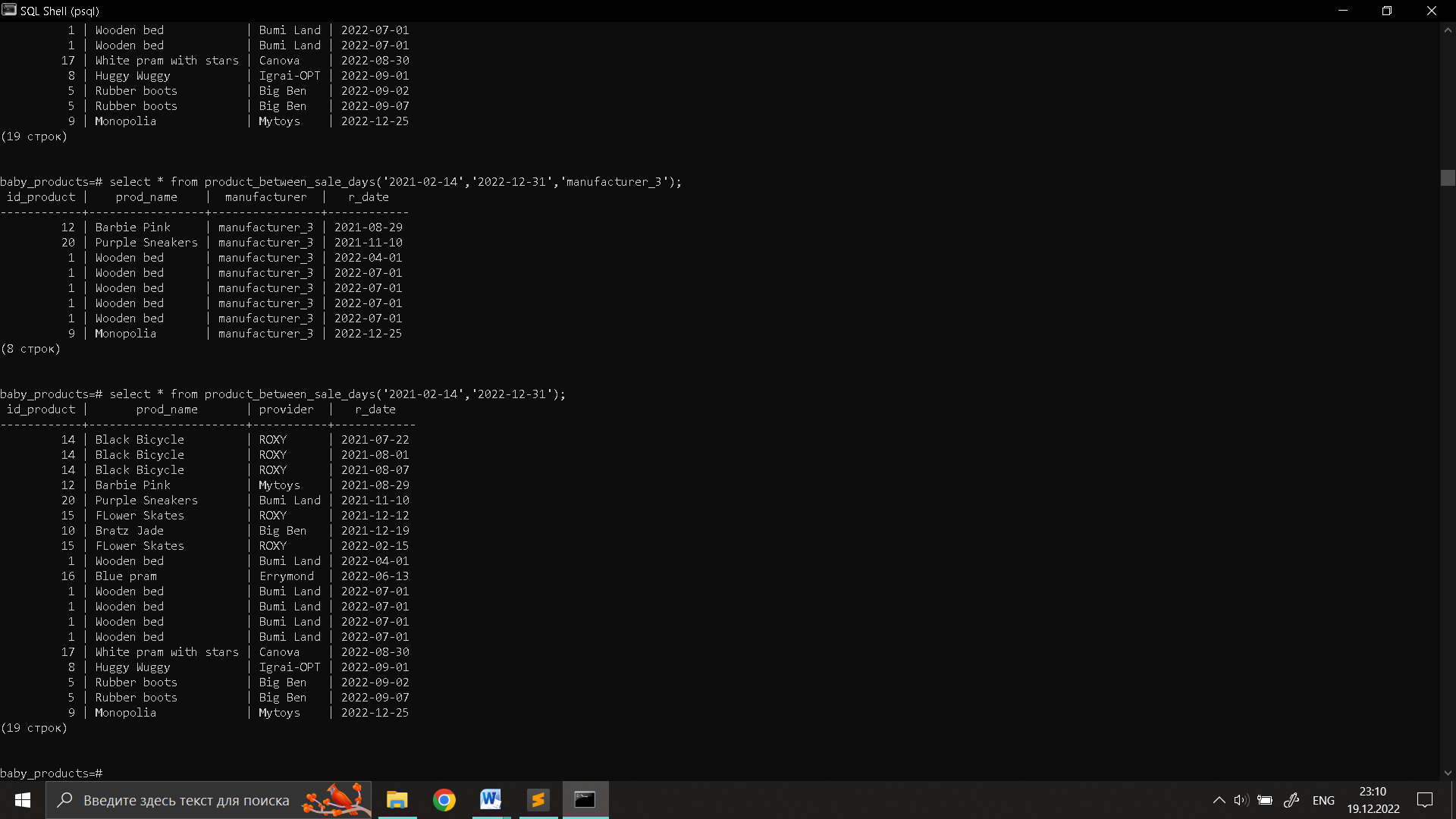
and sales.date\_of\_sale<=b

order by sales.date\_of\_sale;

end;

$$ language plpgsql;

select \* from product\_between\_sale\_days('2021-02-14','2022-12-31');



*Рисунок 22 – Результат работы перегруженной функции.*

1. Найти долю детских товаров, проданных за определенный период (ввод периода) от общего времени продажи.

**Текст запроса:**

create or replace function part\_of\_products\_sales\_days(a date, b date)

returns table(id\_product int,prod\_name text,quantity\_product\_sales bigint , part float)

as $$

declare all\_prod int;

begin

all\_prod = (select sum (sales.quantity) from sales)::bigint::int;

return query select baby\_prod.id\_product ,baby\_prod.prod\_name, sum (sales.quantity) , sum (sales.quantity)/all\_prod::bigint::float from baby\_prod

join sales on baby\_prod.id\_product = sales.id\_product

and sales.date\_of\_sale <= b

and sales.date\_of\_sale >= a

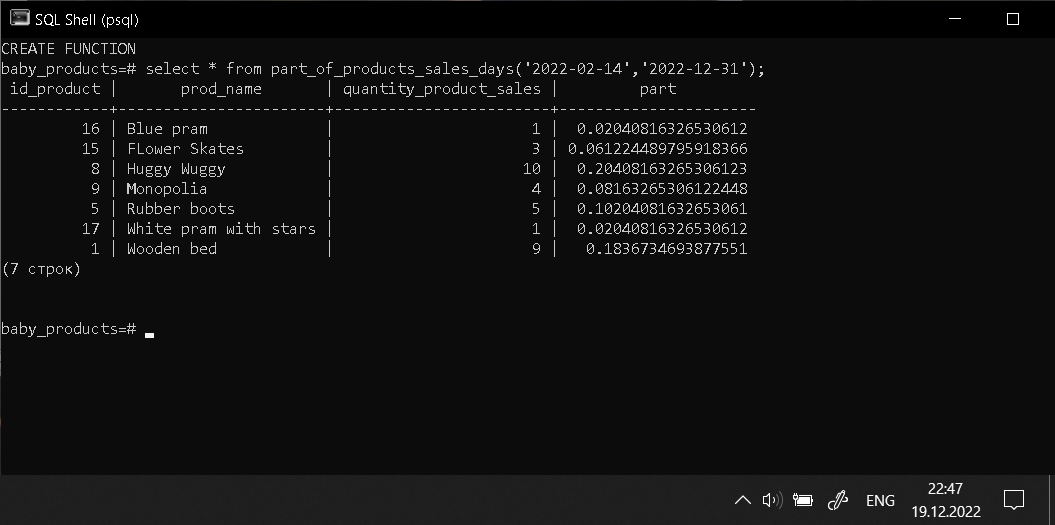
group by baby\_prod.id\_product

order by baby\_prod.prod\_name ;

end;

$$ language plpgsql;

select \* from part\_of\_products\_sales\_days('2022-02-14','2022-12-31');



*Рисунок 23 – Результат запроса.*

1. Найти долю дешевых игрушек (меньше заданного, ввод ограничения) от общего числа игрушек (для каждого возраста (ввод) и в целом).

* Создадим перегруженную функцию, чтобы функция выводила долю дешевых игрушек при вводе возраста для определенного возраста, иначе для каждого возраста.

**Текст запроса:**

create or replace function part\_of\_cheap\_toys(pri real)

returns table ( min\_age int, count\_of\_toys bigint, part float)

as $$

declare all\_count bigint;

BEGIN

all\_count = (select count(\*) from baby\_prod where id\_type=4);

return query select baby\_prod.min\_age, count (id\_product), count(id\_product)/all\_count::bigint::float as part

from baby\_prod where id\_type=4 and baby\_prod.price::numeric::int <= pri

group by baby\_prod.min\_age

order by baby\_prod.min\_age;

END; $$ language plpgsql;

select \* from part\_of\_cheap\_toys(10000);

create or replace function part\_of\_cheap\_toys(age int,pri real)

returns table ( min\_age int, count\_of\_toys bigint, part float)

as $$

declare all\_count bigint;

BEGIN

all\_count = (select count(\*) from baby\_prod where id\_type=4);

return query select baby\_prod.min\_age, count (id\_product), count(id\_product)/all\_count::bigint::float as part

from baby\_prod

where id\_type=4 and baby\_prod.price::numeric::int <= pri

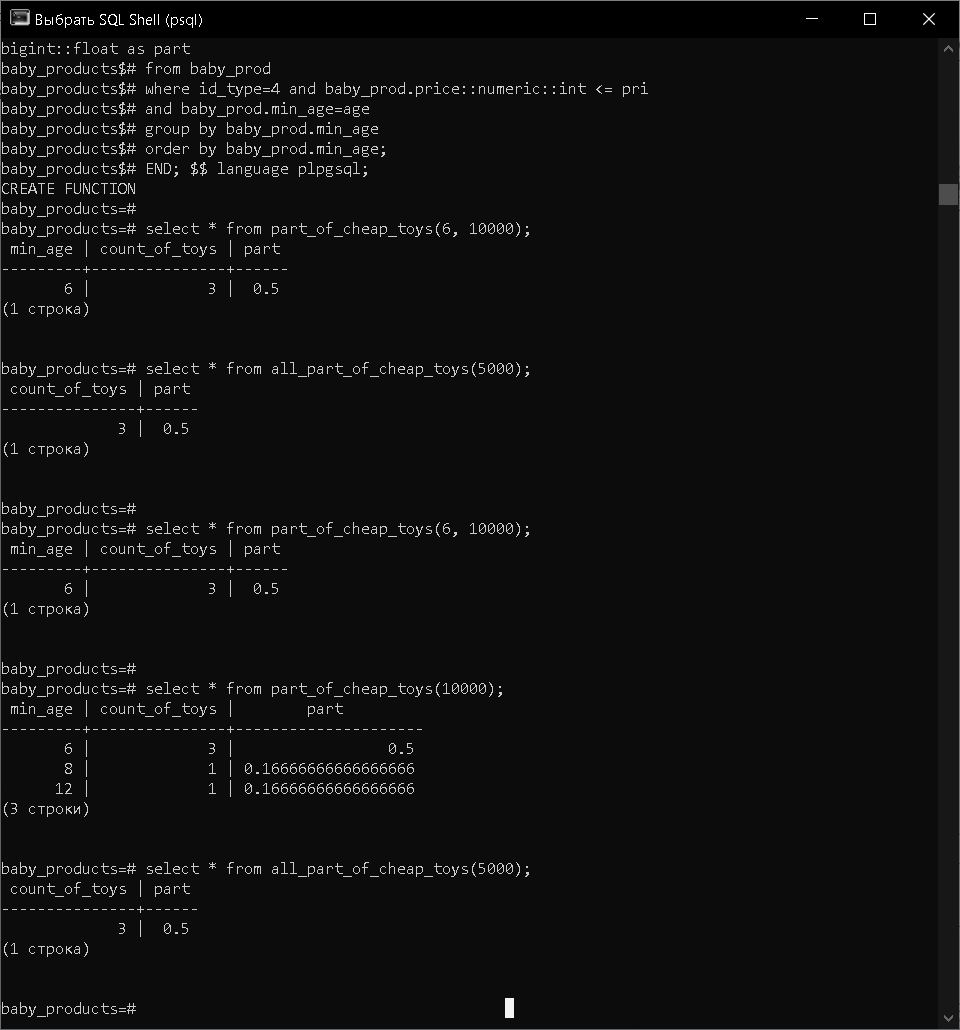
and baby\_prod.min\_age=age

group by baby\_prod.min\_age

order by baby\_prod.min\_age;

END; $$ language plpgsql;

select \* from part\_of\_cheap\_toys(6, 10000);



*Рисунок 24 – Результат запроса.*

* Вывод доли дешевых игрушек от общего числа игрушек в целом.

**Текст запроса:**

create or replace function all\_part\_of\_cheap\_toys(pri real)

returns table ( count\_of\_toys bigint, part float)

as $$

declare all\_count bigint;

BEGIN

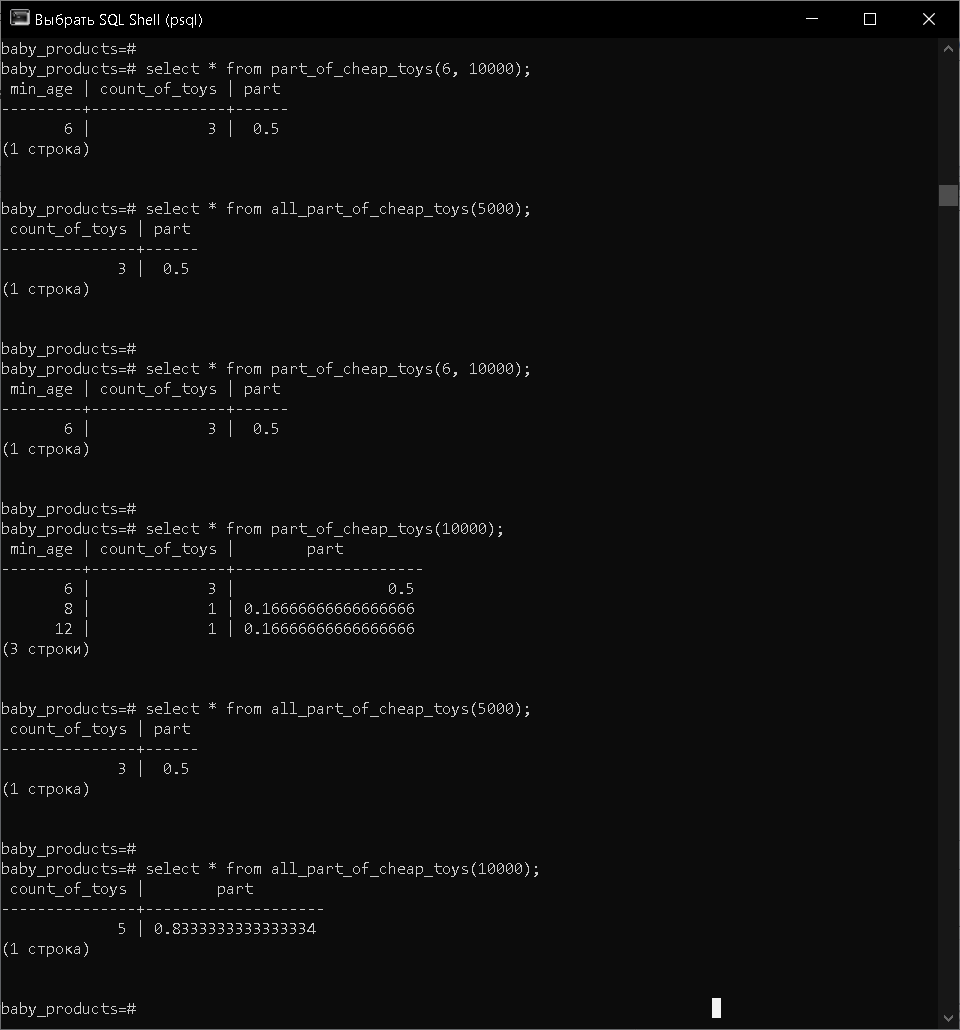
all\_count = (select count(\*) from baby\_prod where id\_type=4);

return query select count(\*), count(\*)/all\_count::bigint::float as part

from baby\_prod where id\_type=4 and baby\_prod.price::numeric::int <= pri;

END; $$ language plpgsql;

select \* from all\_part\_of\_cheap\_toys(10000);



*Рисунок 25 – Результат запроса.*

1. Найти все детские товары, поступившие от заданного поставщика (выбор поставщика), чья стоимость больше, чем средняя стоимость детских товаров, поступивших из заданной страны (выбор страны).

**Текст запроса:**

create or replace function provider\_of\_product\_by\_name(pr\_name text, country text)

returns table(provider text,id\_product int, prod\_name text, price\_of\_prod real, avg\_price\_countryprod real)

as $$

declare

avg\_ real;

begin

avg\_=(select avg(baby\_prod.price) from baby\_prod join providers on baby\_prod.id\_provider=providers.id\_provider

join countries on providers.id\_country=countries.id\_country where countries.country\_name=country);

return query select providers.provider\_name, baby\_prod. id\_product, baby\_prod.prod\_name, baby\_prod.price, avg\_

from baby\_prod

join providers on baby\_prod.id\_provider=providers.id\_provider

where baby\_prod.price>=(avg\_)

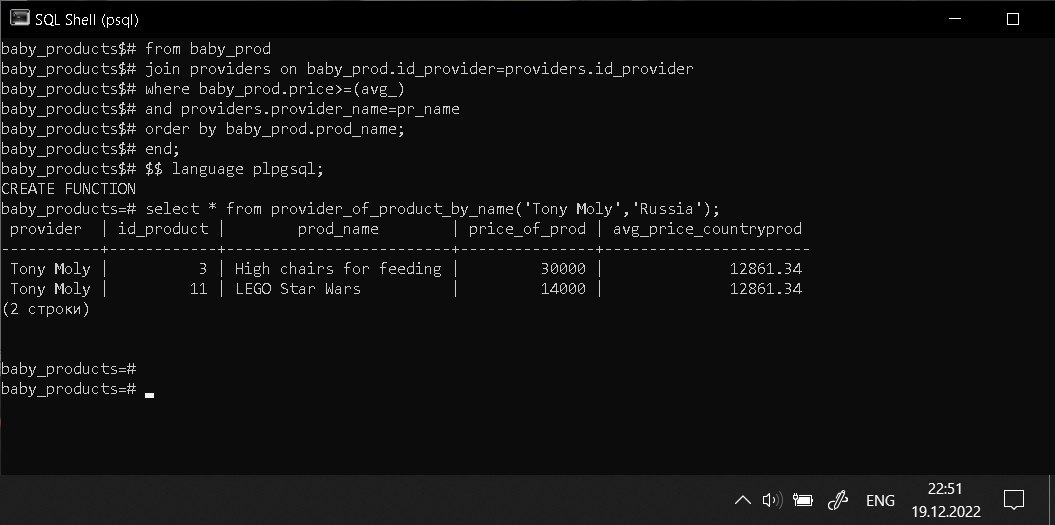
and providers.provider\_name=pr\_name

order by baby\_prod.prod\_name;

end;

$$ language plpgsql;

select \* from provider\_of\_product\_by\_name('Tony Moly','Russia');



*Рисунок 26 – Результат запроса.*

1. Найти долю дешевых детских товаров (чья стоимость меньше заданной, ввод стоимости), поступивших от заданного поставщика (выбор поставщика) и в целом.

**Текст запроса:**

create or replace function providers\_cheap\_products(pri real, provider text)

returns table ( id\_provider int, provider\_name text, count bigint, part float)

as $$

declare all\_count bigint;

BEGIN

all\_count = (select count(\*) from baby\_prod join providers on baby\_prod.id\_provider=providers.id\_provider where providers.provider\_name=provider);

return query select providers.id\_provider, providers.provider\_name, count(id\_product), count(id\_product)/all\_count::bigint::float

from baby\_prod

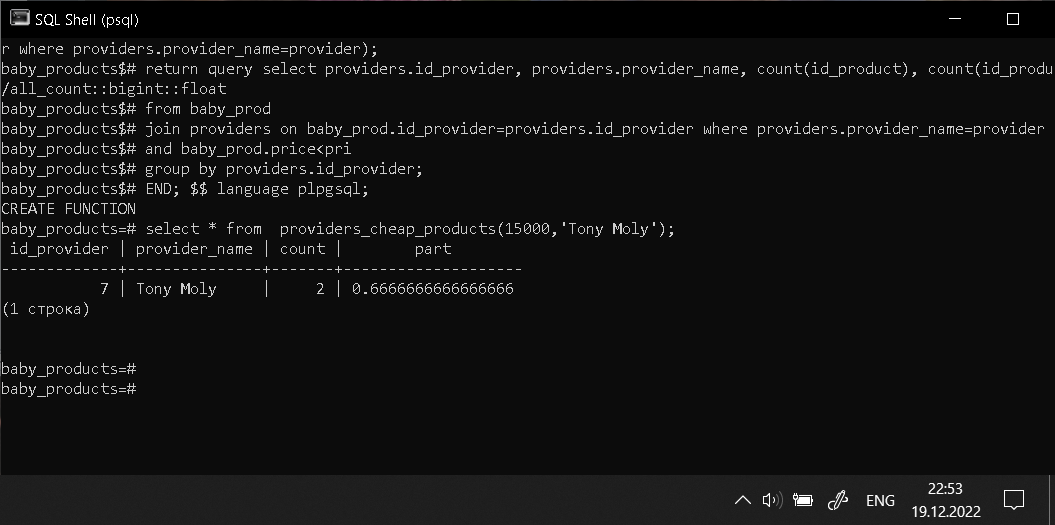
join providers on baby\_prod.id\_provider=providers.id\_provider where providers.provider\_name=provider

and baby\_prod.price<pri

group by providers.id\_provider;

END; $$ language plpgsql;

select \* from providers\_cheap\_products(15000,'Tony Moly');



*Рисунок 27 – Результат запроса.*

1. Найти среднюю стоимость детских товаров, проданных за определенный промежуток времени (ввод интервала).

**Текст запроса:**

create or replace function avg\_price\_in\_period(a date, b date)

returns table (avg\_price float) as $$

begin

return query select avg(baby\_prod.price) from baby\_prod

join sales on baby\_prod.id\_product=sales.id\_product

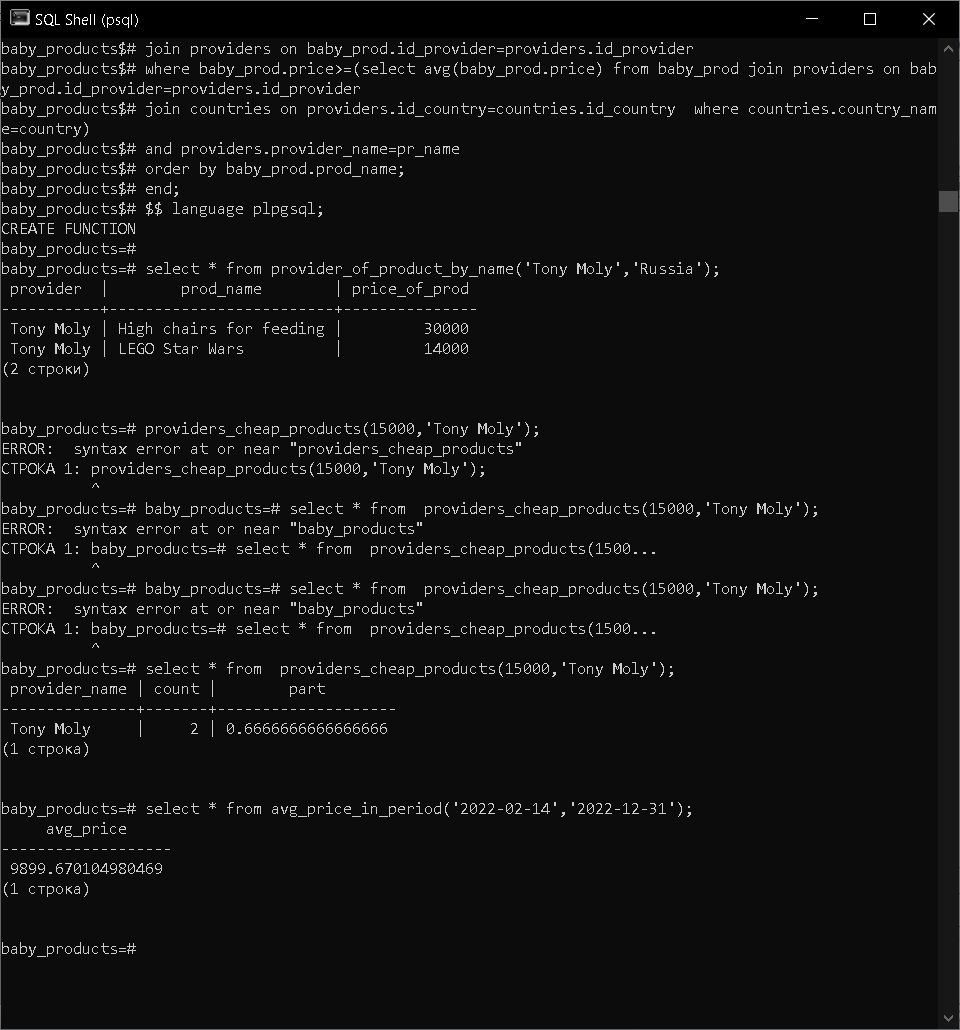
where sales.date\_of\_sale>=a

and sales.date\_of\_sale<=b;

end;

$$ language plpgsql;

select \* from avg\_price\_in\_period('2022-02-14','2022-12-31');



*Рисунок 28 – Результат запроса.*

1. Найти все детские товары, чья стоимость выше, чем средняя стоимость детских товаров заданного производителя (выбор производителя).

**Текст запроса:**

create or replace function price\_of\_product\_more\_than\_avg(man text)

returns table (id\_product int,product\_name text, price real, avg\_price\_of\_product real)

as $$

declare

avg\_ real;

begin

avg\_=(select avg(baby\_prod.price) from baby\_prod

join providers on baby\_prod.id\_provider=providers.id\_provider

join manufacturers on providers.id\_manufact=manufacturers.id\_manufact

where manufacturers.name\_m=man);

return query select baby\_prod.id\_product,baby\_prod.prod\_name, baby\_prod.price, avg\_

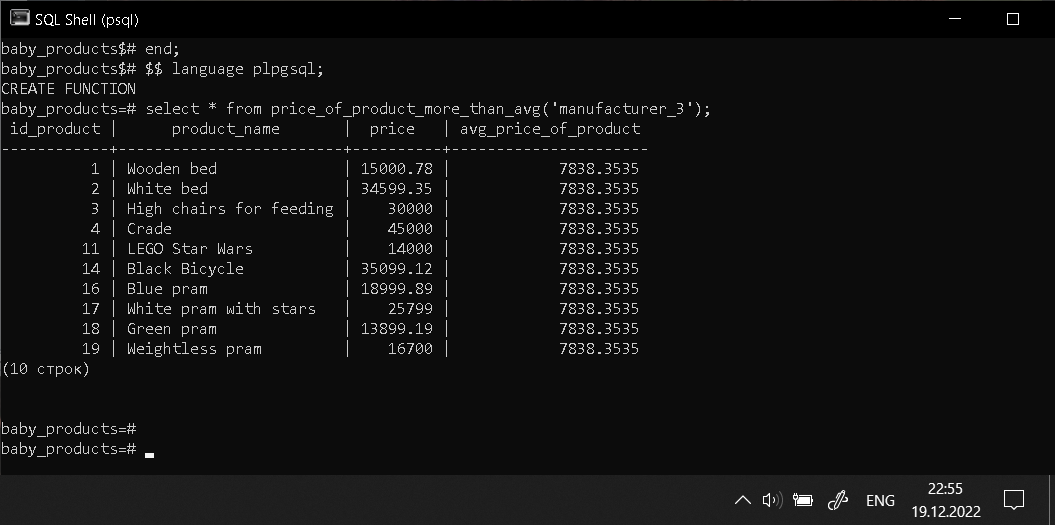
from baby\_prod

where baby\_prod.price>avg\_;

end;

$$ language plpgsql;

select \* from price\_of\_product\_more\_than\_avg('manufacturer\_3');



*Рисунок 29 – Результат запроса.*

1. Найти детские товары, которые лучше всего раскупают, указать параметры (стоимость, производитель, возраст и т.д.).

* Создадим перегруженную функцию, чтобы можно было вводить параметры по отдельности.

**Текст запроса:**

create or replace function popular\_baby\_prod(pri real , man text, age int)

returns table (id\_product int, prod\_name text, product\_price real, manufacturer\_name text, min\_age int, sale\_quant bigint) as $$

begin

return query select baby\_prod.id\_product ,baby\_prod.prod\_name, baby\_prod.price, manufacturers.name\_m, baby\_prod.min\_age, (select sum(sales.quantity) from sales where sales.id\_product=baby\_prod.id\_product) from baby\_prod

join providers on baby\_prod.id\_provider=providers.id\_provider

join manufacturers on providers.id\_manufact=manufacturers.id\_manufact

join sales on baby\_prod.id\_product=sales.id\_product

where

baby\_prod.price=pri

and manufacturers.name\_m=man

and baby\_prod.min\_age=age

group by baby\_prod.id\_product, manufacturers.id\_manufact

order by sale\_quant desc limit 3;

end; $$

language 'plpgsql';

select \* from popular\_baby\_prod(5000,'manufacturer\_3',12);

create or replace function popular\_baby\_prod(man text)

returns table (id\_product int, prod\_name text, product\_price real, manufacturer\_name text, min\_age int, sale\_quant bigint) as $$

begin

return query select baby\_prod.id\_product ,baby\_prod.prod\_name, baby\_prod.price, manufacturers.name\_m, baby\_prod.min\_age, (select sum(sales.quantity) from sales where sales.id\_product=baby\_prod.id\_product) from baby\_prod

join providers on baby\_prod.id\_provider=providers.id\_provider

join manufacturers on providers.id\_manufact=manufacturers.id\_manufact

join sales on baby\_prod.id\_product=sales.id\_product

where manufacturers.name\_m=man

group by baby\_prod.id\_product, manufacturers.id\_manufact

order by sale\_quant desc limit 3;

end; $$

language 'plpgsql';

select \* from popular\_baby\_prod('manufacturer\_3');

create or replace function popular\_baby\_prod(age int)

returns table (id\_product int, prod\_name text, product\_price real, manufacturer\_name text, min\_age int, sale\_quant bigint) as $$

begin

return query select baby\_prod.id\_product ,baby\_prod.prod\_name, baby\_prod.price, manufacturers.name\_m, baby\_prod.min\_age, (select sum(sales.quantity) from sales where sales.id\_product=baby\_prod.id\_product) from baby\_prod

join providers on baby\_prod.id\_provider=providers.id\_provider

join manufacturers on providers.id\_manufact=manufacturers.id\_manufact

join sales on baby\_prod.id\_product=sales.id\_product

where baby\_prod.min\_age=age

group by baby\_prod.id\_product, manufacturers.id\_manufact

order by sale\_quant desc limit 3;

end; $$

language 'plpgsql';

select \* from popular\_baby\_prod(12);

create or replace function popular\_baby\_prod(pri real)

returns table (id\_product int, prod\_name text, product\_price real, manufacturer\_name text, min\_age int, sale\_quant bigint) as $$

begin

return query select baby\_prod.id\_product ,baby\_prod.prod\_name, baby\_prod.price, manufacturers.name\_m, baby\_prod.min\_age, (select sum(sales.quantity) from sales where sales.id\_product=baby\_prod.id\_product) from baby\_prod

join providers on baby\_prod.id\_provider=providers.id\_provider

join manufacturers on providers.id\_manufact=manufacturers.id\_manufact

join sales on baby\_prod.id\_product=sales.id\_product

where baby\_prod.price=pri

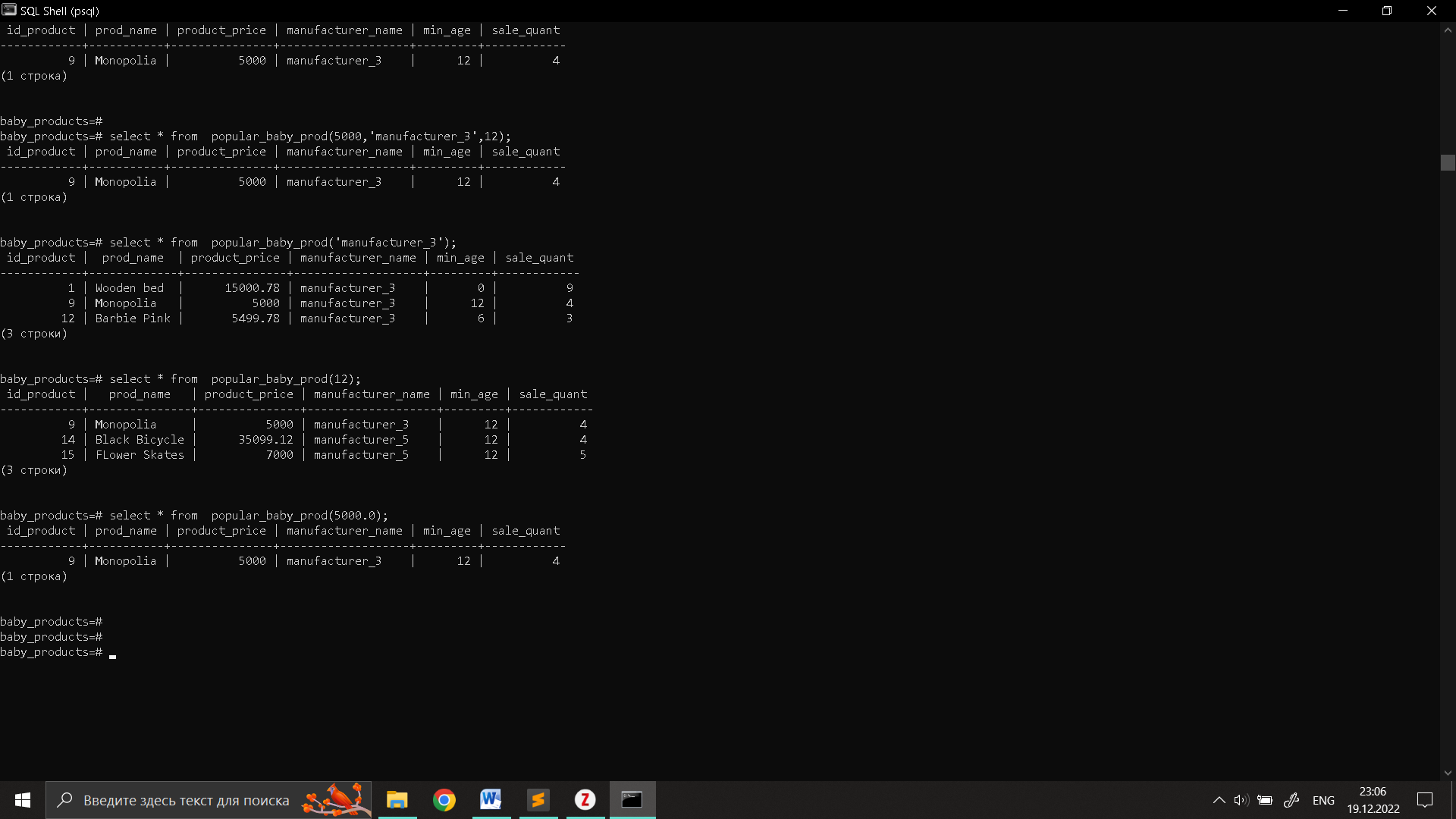
group by baby\_prod.id\_product, manufacturers.id\_manufact

order by sale\_quant desc limit 3;

end; $$

language 'plpgsql';

select \* from popular\_baby\_prod(5000.0);



*Рисунок 30 – Результат запроса.*

**8. Анализ результатов и выводы**

Построена база данных в СУБД PostgreSQL. В базе данных присутствуют таблиц-справочников и таблиц, использующих справочники. В пояснительной записке представлена структура таблиц и связи между ними.

Написаны функции для реализации запросов с использованием процедурного языка plpgsql, приведены примеры результатов выполнения функций. Для реализации курсовой работы были применены знания и навыки разработки БД, выполнения запросов, написания функций, полученные в ходе выполнения лабораторных работ по курсу и изучения документации PostgreSQL.

**9. Использованные источники**

1. PostgreSQL [Электронный ресурс]: Документация по PostgreSQL. – Режим доступа: http://postgresql.ru.net . - загл. с экрана. (Дата обращения: 10.12.2022)

1. Трошина Г. В. Базы данных [Электронный ресурс] : электронный учебно-методический комплекс / Г. В. Трошина ; Новосиб. гос. техн. ун-т. - Новосибирск, [2011]. - Режим доступа: http://elibrary.nstu.ru/source?bib\_id=vtls000152242. - Загл. с экрана – (Дата обращения : 10.11.2022)