МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет автоматики и вычислительной техники

Кафедра вычислительной техники



Курсовая работа

По дисциплине: «Базы данных»

Разработка базы данных «Полиграфическая продукция»

|  |  |
| --- | --- |
| Факультет: АВТ  Группа: АВТ-008  Студент: Седаков И.В. | Преподаватель: Трошина Г.В. |

Новосибирск

2022 г.

**Содержание**

[***Введение 3***](#_Toc121180875)

[***Задание 3***](#_Toc121180876)

[***Графическое представление связей между таблицами 5***](#_Toc121180877)

[***Структура таблиц базы данных 5***](#_Toc121180878)

[***Содержание таблиц базы данных 7***](#_Toc121180879)

[***Руководство пользователя 9***](#_Toc121180880)

[***Тексты запросов и примеры работы 12***](#_Toc121180881)

[***Проверка работы ролей 33***](#_Toc121180882)

[***Анализ результатов и выводы 34***](#_Toc121180883)

[***Список использованных источников 35***](#_Toc121180884)

[***Листинг функций, используемых в запросах 36***](#_Toc121180885)

[***Создание ролей БД 46***](#_Toc121180886)

[***Листинг создания и заполнения таблиц БД 47***](#_Toc121180887)

# Введение

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД).

В ходе выполнения курсовой работы необходимо разработать базу данных «Полиграфическая продукция», сформировав несколько таблиц и реализовать запросы согласно варианту. Также необходимо предусмотреть наличие ролей управления базой данных.

Цель работы: закрепление навыков разработки и проектирования баз данных, а также написания функций, реализующих управление и обработку запросов.

Актуальность работы: современным торговым точкам требуются базы данных для учета товаров, в данной работе был продемонстрирован пример базы данных, которая может быть использована в сети магазинов, занимающихся продажей различной полиграфической продукцией. К тому же, умение работать с базами данных необходимо во многих областях разработки программного обеспечения, поэтому актуальность этой работы высокая.

# Задание

Разработать базу данных «Полиграфическая продукция»

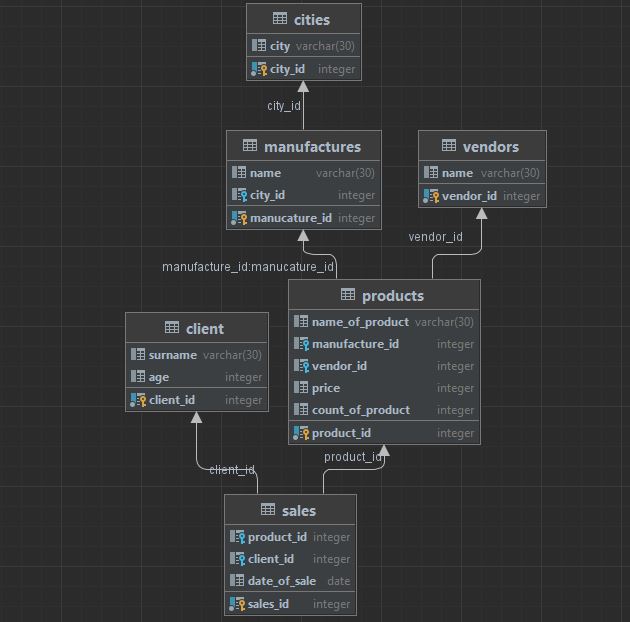
Предусмотреть: ввод данных, редактирование, просмотр данных. Обязательные требования к базе данных:

* Наличие таблиц-справочников и таблиц, использующих справочники;
* Предусмотреть следующие роли:
  + Оператор БД
  + Пользователь БД
  + Администратор БД

Реализовать следующие типы запросов с использованием нескольких таблиц:

* Для каждого вида полиграфической продукции указать сведения о ней( наименование, издательство, цена, тираж и т.п)
* Для каждого вида полиграфической продукции получить список, отсортированный: в алфавитном порядке по наименованию, по тиражу, по стоимости
* Найти самую дорогую полиграфическую продукцию, самую дешевую, среднюю стоимость.
* Найти всю полиграфическую продукцию, с ценой выше заданной(ввод)
* Найти всю полиграфическую продукцию, чей тираж попадает в заданные интервалы(ввод).
* Для заданного издательства(ввод) найти всю полиграфическую продукцию.
* Найти всю полиграфическую продукцию, чья цена попадает в заданные интервалы(ввод) и для заданного издательства.
* Найти долю дешевой полиграфической продукции(ввод).
* Найти долю полиграфической продукции, проданной за определенный период(ввод интервала).
* Найти всю полиграфическую продукцию, поступившую от заданного издателя(ввод), чья стоимость выше заданной(ввод).
* Найти всю полиграфическую продукцию, поступившую от заданного издательства(ввод), чья стоимость больше чем средняя стоимость продукции, поступившей из заданного города(ввод).
* Найти среднюю стоимость продукции, проданной за определенный промежуток времени.
* Найти долю дорогой полиграфической продукции, поступившей от заданного производителя(ввод), и в целом
* Найти среднюю стоимость продукции, проданной за определенный промежуток времени(ввод интервала).
* Найти всю полиграфическую продукцию, чья стоимость выше, чем средняя стоимость полиграфической продукции заданного издательства(ввод).
* Найти всю полиграфическую продукцию, которую лучше всего раскупают(стоимость, издательство и тп.).
* Найти долю полиграфической продукции для заданного возраста(например для детей, для младшего возраста, для старшего школьного возраста и тп.) от общего числа полиграфической продукции.

# Графическое представление связей между таблицами



*Рис. 1. Графическое представление связей между таблицами.*

# Структура таблиц базы данных

Таблица-справочник *cities* содержит информацию о странах*:*

* city\_id: serial – идентификатор города
* city: varchar (30) – название города

Таблица-справочник *vendors* содержит информацию о поставщиках*:*

* vendor\_id: serial – идентификатор поставщика
* name: varchar (30) – наименование поставщика

Таблица-справочник *client* содержит информацию о клиентах*:*

* client\_id: serial – идентификатор клиента
* surname: varchar (30) – фамилия клиента
* age: integer – возраст покупателя

Таблица-справочник *manufactures* содержит информацию о производителях*:*

* manufactures\_id: serial – идентификатор производителя
* name: varchar (30) – наименование производителя
* city\_id: integer – ссылка на идентификатор города

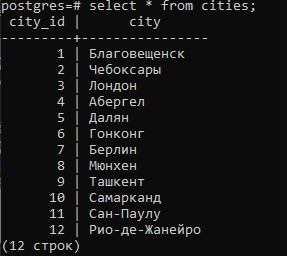
Таблица *products* содержит информацию о полиграфической продукции*:*

* product\_id: serial – идентификатор товара
* name\_of\_product: varchar (30) – наименование товара
* vendor\_id: integer – идентификатор вендора
* manufacture\_id: integer – идентификатор производителя
* price: integer – стоимость товара
* count\_of\_product: integer – тираж

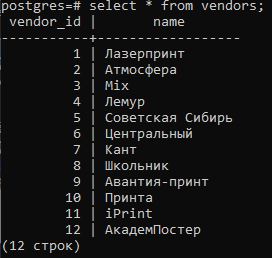
Таблица *sales* содержит информацию о продажах полиграфической продукции*:*

* sales\_id: serial – идентификатор продажи товара
* products\_id: integer – ссылка на идентификатор товара
* client\_id: integer – ссылка на идентификатор клиента
* date\_of\_sale: date – дата продажи товара

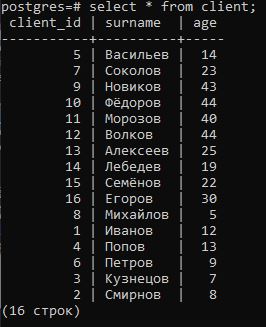
# Содержание таблиц базы данных



*Рис. 2. Содержимое таблицы с городами.*



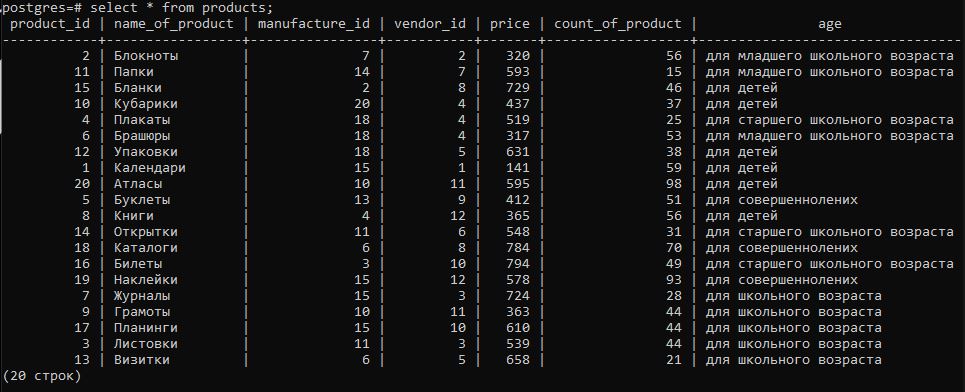
*Рис. 3. Содержимое таблицы с поставщиками.*



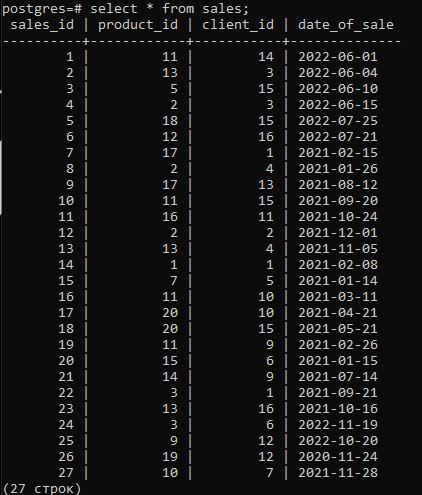
*Рис. 4. Содержимое таблицы с клиентами.*



*Рис. 5. Содержимое таблицы с производителями.*



*Рис. 6. Содержимое таблицы с продукцией.*



*Рис. 7. Содержимое таблицы с данными о продажах.*

# Руководство пользователя

*Таблица 1. Описание функций*

|  |  |
| --- | --- |
| Сигнатура функции | Описание функции |
| *products\_info()* | Для каждого вида полиграфической продукции выводит сведения о ней |
| products\_info\_ordered\_by\_name() | Выводит отсортированный список по наименованию |
| products\_info\_ordered\_by\_price() | Выводит отсортированный список по цене |
| products\_info\_ordered\_by\_count() | Выводит отсортированный список по тиражу |
| max\_min\_avg\_price() | Выводит максимальную, минимальную, среднюю цену продуктов |
|  |  |
| *products\_price\_more\_then (a integer)* | Выводит всю полиграфическую продукцию, с ценой выше заданной |
| *products\_count\_between (a integer, b integer)* | Выводи всю полиграфическую продукцию, чей тираж попадает в заданные интервалы |
| *products\_by\_manufactures (\_manufact text)* | Выводит для заданного издательства всю полиграфическую продукцию |
| *products\_price\_between\_and\_manufactures (a integer, b integer, name text)* | Выводит всю полиграфическую продукцию, чья цена попадает в заданные интервалы и для заданного издательства |
| *products\_percentage\_price\_less\_then (a integer)* | Выводит долю дешевой полиграфической продукции |
| *products\_percentage\_data\_between (a date, b date)* | Выводит долю полиграфической продукции, проданной за определенный период |
| *products\_price\_more\_then\_and\_manufactures (a integer, name text)* | Выводит всю полиграфическую продукцию, поступившую от заданного издателя, чья стоимость выше заданной |
| *products\_manufacture\_city (\_manufact text, \_city text)* | Выводит всю полиграфическую продукцию, поступившую от заданного издательства, чья стоимость больше чем средняя стоимость продукции, поступившей из заданного города |
| *products\_percentage\_price\_more\_then\_and\_manufactures (\_price integer, \_manufact text)* | Выводит долю дорогой полиграфической продукции, поступившей от заданного производителя, и в целом |
| *products\_avg\_data\_between (date1 date, date2 date)* | Выводит среднюю стоимость продукции, проданной за определенный промежуток времени |
| *products\_price\_more\_then\_avg\_in\_manufactures (manufact text)* | Выводит всю полиграфическую продукцию, чья стоимость выше, чем средняя стоимость полиграфической продукции заданного издательства |
| *best\_saled\_products ()* | Выводит всю полиграфическую продукцию, которую лучше всего раскупают |
| *products\_percentage\_age(\_age text)* | Выводит долю полиграфической продукции для заданного возраста от общего числа полиграфической продукции |

# Тексты запросов и примеры работы

* + 1. Для каждого вида полиграфической продукции указать сведения о ней

(наименование, издательство, цена, тираж, и т.п.).

create or replace function products\_info()

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer

)

as

$$

begin

return query select

products.product\_id,

products.name\_of\_product,

products.manufacture\_id,

products.vendor\_id,

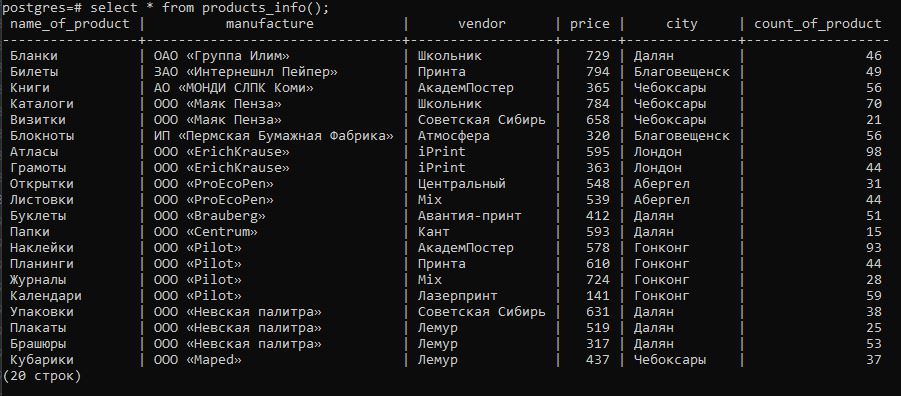
products.price,

products.count\_of\_product

from products;

end;

$$ language plpgsql;



*Рис. 8. Сведения о каждом экземпляре продукта.*

* + 1. Для каждого вида полиграфической продукции получить список, отсортированный: в алфавитном порядке по наименованию, по тиражу, по стоимости.

Список, отсортированный по названию:

create or replace function products\_info\_ordered\_by\_name()

returns table(

name varchar(30),

price integer,

count integer,

age varchar(100)

)

as

$$

begin

return query select products.name\_of\_product,

products.price,

products.count\_of\_product,

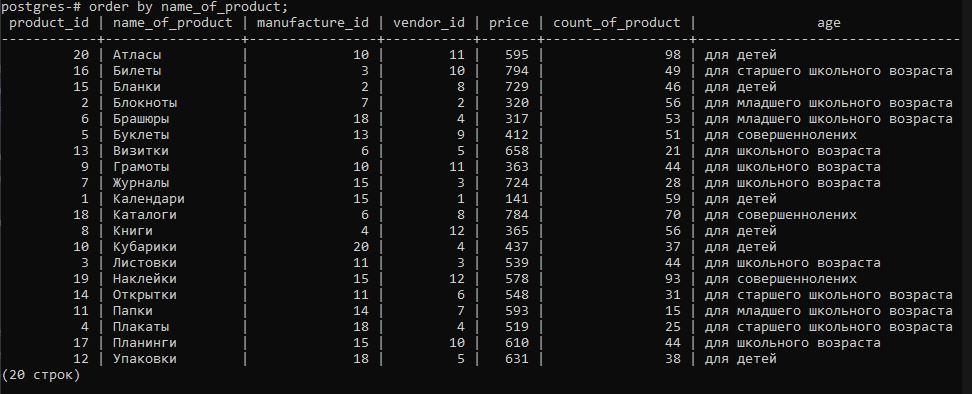
products.age

from products

order by products.name\_of\_product;

end;

$$ language plpgsql;



*Рис. 9. Список, отсортированный в алфавитном порядке по названию.*

Список, отсортированный в алфавитном порядке по тиражу:

create or replace function products\_info\_ordered\_by\_count()

returns table(

name varchar(30),

price integer,

count integer,

age varchar(100)

)

as

$$

begin

return query select products.name\_of\_product,

products.price,

products.count\_of\_product,

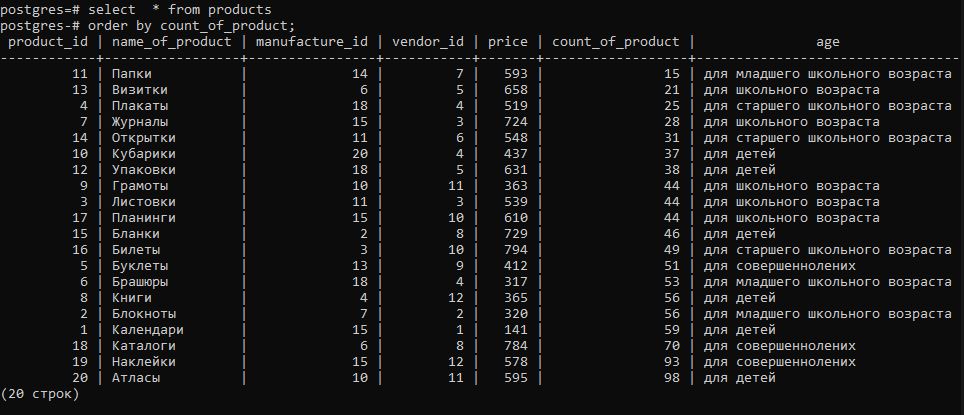
products.age

from products

order by products.count\_of\_product;

end;

$$ language plpgsql;



*Рис. 10. Список, отсортированный по тиражу.*

Список, отсортированный по стоимости:

create or replace function products\_info\_ordered\_by\_price()

returns table(

name varchar(30),

price integer,

count integer,

age varchar(100)

)

as

$$

begin

return query select products.name\_of\_product,

products.price,

products.count\_of\_product,

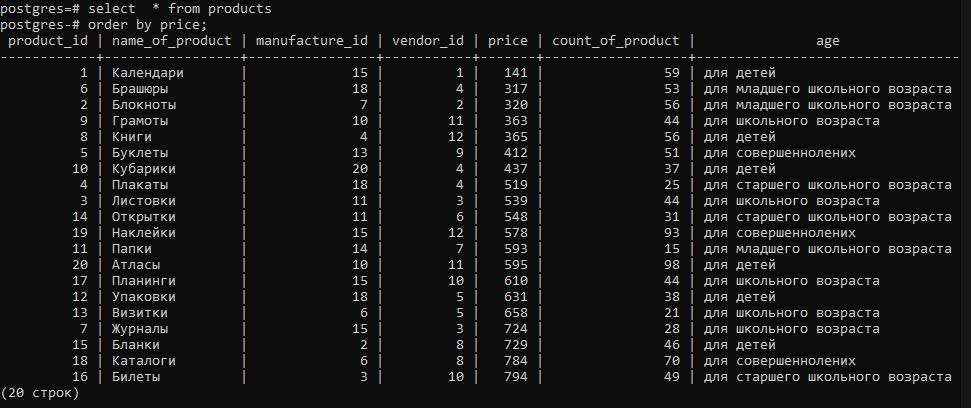
products.age

from products

order by products.price;

end;

$$ language plpgsql;



*Рис. 11. Список, отсортированный по стоимости.*

* + 1. Найти самую дорогую полиграфическую продукцию, самую дешевую, среднюю стоимость.

create or replace function max\_min\_avg\_price()

returns table(

max\_price integer,

min\_price integer,

avg\_price integer

)

as

$$

begin

return query select max(products.price)::integer,

min(products.price)::integer,

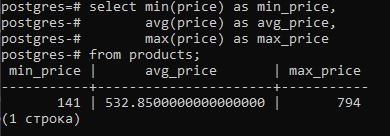
avg(products.price)::integer

from products;

end;

$$

language plpgsql;



*Рис. 12. Минимальная цена, средняя цена, максимальная цена на продукцию.*

* + 1. Найти всю полиграфическую продукцию, с ценой выше заданной.

create or replace function products\_price\_more\_then(a integer)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer

)

as

$$

declare

\_a integer;

begin

select a into \_a;

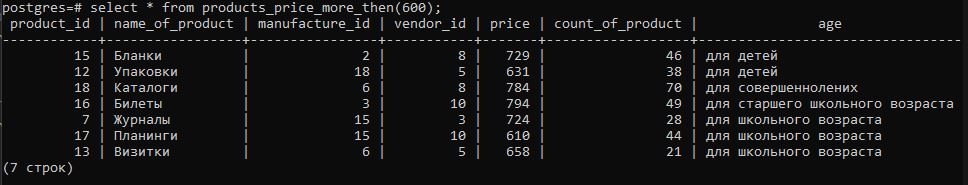
return query select \*

from products

where products.price > \_a;

end;

$$ language plpgsql;



*Рис. 13. Продукция с ценой выше 600.*

* + 1. Найти всю полиграфическую продукцию, чей тираж попадает в заданные интервалы(ввод).

create or replace function products\_count\_between(a integer, b integer)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer,

age varchar(100)

)

as

$$

declare

\_a integer;

\_b integer;

begin

select a into \_a;

select b into \_b;

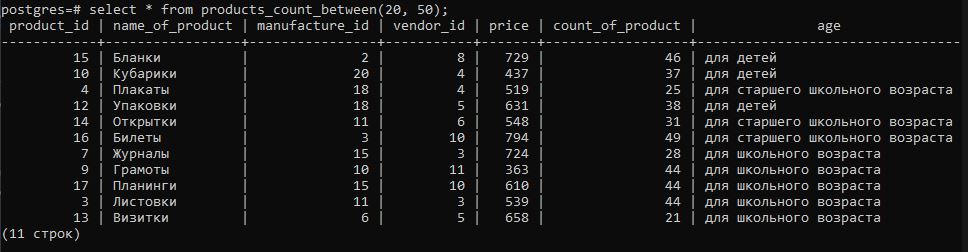
return query select \*

from products

where products.count\_of\_product between a and b;

end;

$$ language plpgsql;

*Рис. 14. Вся полиграфическая продукция тираж которой составляет 20 до 50 экземпляров*

* + 1. Для заданного издательства(выбор) найти всю полиграфическую продукцию.

create or replace function products\_by\_manufactures(\_manufact text)

returns table(

name\_of\_product varchar(30),

price integer,

count\_of\_product integer

)

as

$$

begin

return query select

products.name\_of\_product,

products.price,

products.count\_of\_product

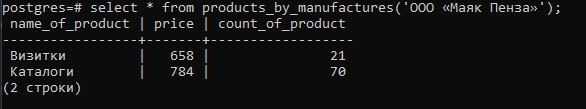
from products

JOIN manufactures m on products.manufacture\_id = m.manucature\_id

where m.name = \_manufact;

end;

$$ language plpgsql;

**

*Рис. 15. Продукция соответствующая производителю Маяк Пенза.*

* + 1. Найти всю полиграфическую продукцию, чья цена попадает в заданные интервалы и для заданного издательства.

create or replace function products\_price\_between\_and\_manufactures(a integer, b integer, name text)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer

)

as

$$

declare

\_a integer;

\_b integer;

\_name text;

begin

select a into \_a;

select b into \_b;

select name into \_name;

return query select

products.product\_id,

products.name\_of\_product,

products.manufacture\_id,

products.vendor\_id,

products.price,

products.count\_of\_product

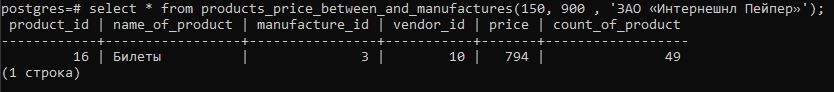
from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

where products.price between a and b and m.name = \_name;

end;

$$ language plpgsql;



*Рис. 16. Полиграфическая продукция, цена которой находится в интервале 150 и 900, которая соответствует производителю Интернешнл Пейпер.*

* + 1. Найти долю дешевой полиграфической продукции.

create or replace function products\_percentage\_price\_less\_then(a integer)

returns integer

as

$$

declare

\_cheap integer;

\_all integer;

begin

select count(name\_of\_product) into \_all from products;

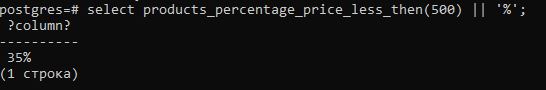
select count(name\_of\_product) into \_cheap from products

where price < a;

return round((\_cheap::double precision / \_all::double precision) \* 100);

end;

$$ language plpgsql;



*Рис. 17.Доля дешевой продукции, цена которой меньше 500.*

* + 1. Найти долю полиграфической продукции, проданной за определенный период.

create or replace function products\_percentage\_data\_between(a date, b date)

returns integer

as

$$

declare

count integer;

\_all integer;

begin

select count(date\_of\_sale) into count from sales

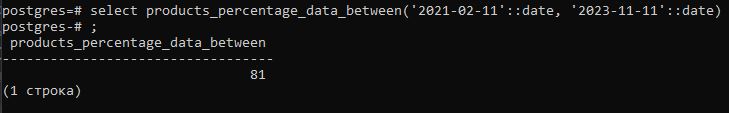
where date\_of\_sale between a and b;

select count(date\_of\_sale) into \_all from sales;

return round((count::double precision / \_all::double precision) \* 100);

end;

$$ language plpgsql;

**

*Рис. 18. Доля продукции, проданная в период с 2022-02-11 по 2023-11-11.*

* + 1. Найти всю полиграфическую продукцию, поступившую от заданного издателя, чья стоимость выше заданной.

create or replace function products\_price\_more\_then\_and\_manufactures(a integer, name text)

returns table

(

name\_of\_product varchar(30),

manufacture varchar(30),

price integer,

count\_of\_product integer

)

as

$$

declare

\_a integer;

\_name text;

begin

select a into \_a;

select name into \_name;

return query select

products.name\_of\_product,

m.name,

products.price,

products.count\_of\_product

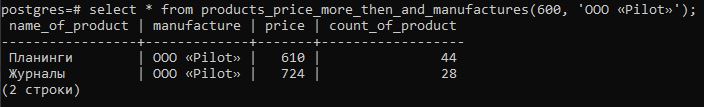
from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

where products.price > a and m.name = \_name;

end;

$$ language plpgsql;



*Рис. 19. Продукция полученная от заданного производителя, чья стоимость выше 600.*

* + 1. Найти всю полиграфическую продукцию, поступившую от заданного издательства, чья стоимость больше чем средняя стоимость продукции, поступившей из заданного города.

create or replace function products\_manufacture\_city(\_manufact text, \_city text)

returns table

(

name\_of\_product varchar(30),

city varchar(30),

price integer,

avg\_price integer,

count\_of\_product integer

)

as

$$

declare

avg\_price integer;

begin

select avg(products.price) into avg\_price from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

JOIN cities as c ON m.city\_id = c.city\_id

where c.city = \_city;

return query select

products.name\_of\_product,

m.name,

products.price,

avg\_price,

products.count\_of\_product

from products

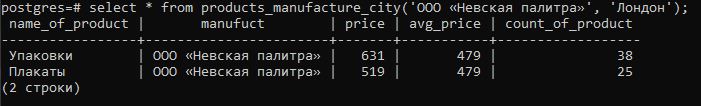
JOIN manufactures m on m.manucature\_id = products.manufacture\_id

JOIN cities as c ON m.city\_id = c.city\_id

where m.name = \_manufact and products.price >avg\_price;

end;

$$ language plpgsql;



*Рис. 20. Продукция, поступившая от Невская Палитра, чья стоимость больше, чем средняя стоимость продукции, поступившей из Лондона.*

* + 1. Найти долю дорогой полиграфической продукции, поступившей от заданного производителя, и в целом.

create or replace function products\_percentage\_price\_more\_then\_and\_manufactures(\_price integer, \_manufact text)

returns table(

expensive integer,

expensive\_all integer

)

as

$$

declare

\_expensive\_manufact integer;

\_expensive integer;

\_all integer;

begin

select count(products.name\_of\_product) into \_expensive\_manufact from products

JOIN manufactures m on products.manufacture\_id = m.manucature\_id

where products.price > \_price and m.name = \_manufact;

select count(products.name\_of\_product) into \_expensive from products

where products.price > \_price;

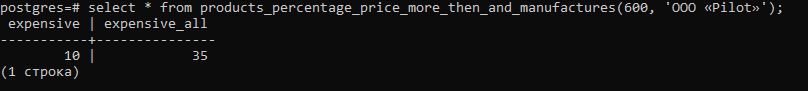
select count(products.name\_of\_product) into \_all from products;

return query select round((\_expensive\_manufact::double precision/\_all::double precision)\*100)::integer,

round((\_expensive::double precision/\_all::double precision)\*100)::integer;

end

$$ language plpgsql;



*Рис. 24. Процент дорогой продукции, поступившей от заданного производителя и в целом.*

* + 1. Найти среднюю стоимость продукции, проданной за определенный промежуток времени.

create or replace function products\_avg\_data\_between(date1 date, date2 date)

returns integer

as

$$

declare

count integer;

begin

select avg(products.price) into count from products

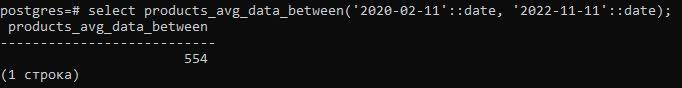
join sales s on products.product\_id = s.product\_id

where date\_of\_sale between date1 and date2;

return count;

end;

$$ language plpgsql;



*Рис. 25. Средняя стоимость продукции, проданной в период с 2022-02-11 по 2022-11-11.*

* + 1. Найти всю полиграфическую продукцию, чья стоимость выше, чем средняя стоимость полиграфической продукции заданного издательства.

create or replace function products\_price\_more\_then\_avg\_in\_manufactures(manufact text)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

avg\_prc integer

)

as

$$

declare

avg\_price integer;

begin

select avg(products.price) into avg\_price from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

where m.name = manufact;

return query select

products.product\_id,

products.name\_of\_product,

products.manufacture\_id,

products.vendor\_id,

products.price,

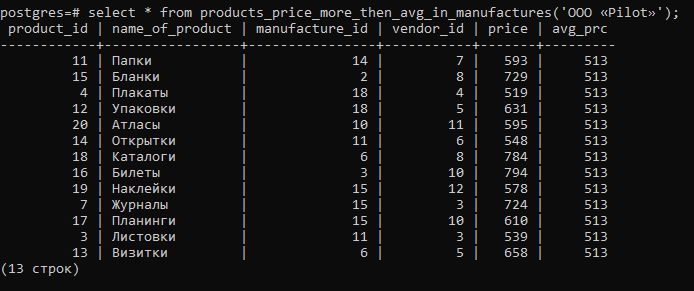
avg\_price

from products

where products.price > avg\_price;

end;

$$ language plpgsql;



*Рис. 26. Товары, чья стоимость выше, чем средняя стоимость полиграфической продукции от ‘OOO Pilot’.*

* + 1. Найти всю полиграфическую продукцию, которую лучше всего раскупают.

create or replace function best\_saled\_products()

returns table

(

product varchar(30),

manufacture varchar(30),

vendor varchar(30),

price integer,

count\_of\_sales integer

)

as

$$

begin

return query select

products.name\_of\_product,

m.name,

v.name,

products.price,

count(products.name\_of\_product)::integer as count\_of\_products

from products

JOIN sales s on products.product\_id = s.product\_id

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

JOIN vendors v on products.vendor\_id = v.vendor\_id

group by s.product\_id,

products.product\_id,

m.manucature\_id,

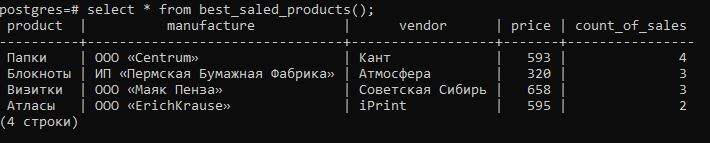
v.vendor\_id

order by count\_of\_products desc

limit 4;

end;

$$ language plpgsql;



*Рис. 27. Продукты, которые лучше всего продаются.*

* + 1. Найти долю полиграфической продукции для заданного возраста

от общего числа полиграфической продукции.

create or replace function products\_percentage\_age(\_age text)

returns integer

as

$$

declare

count integer;

\_all integer;

begin

select count(name\_of\_product) into count from products

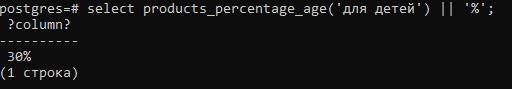
where products.age = \_age;

select count(name\_of\_product) into \_all from products;

return round((count::double precision / \_all::double precision) \* 100);

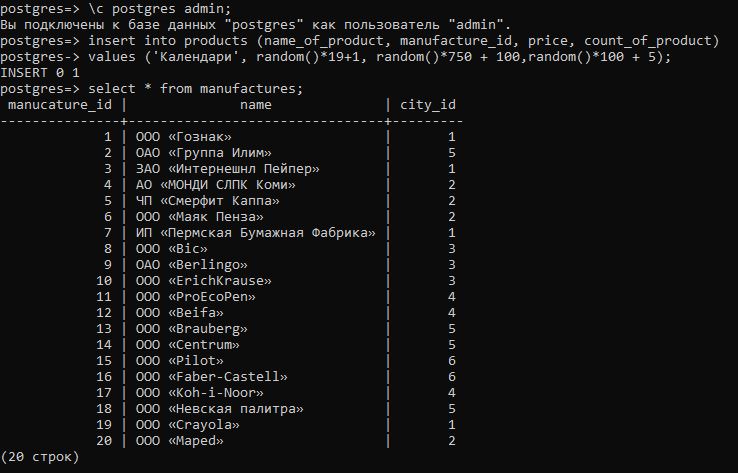
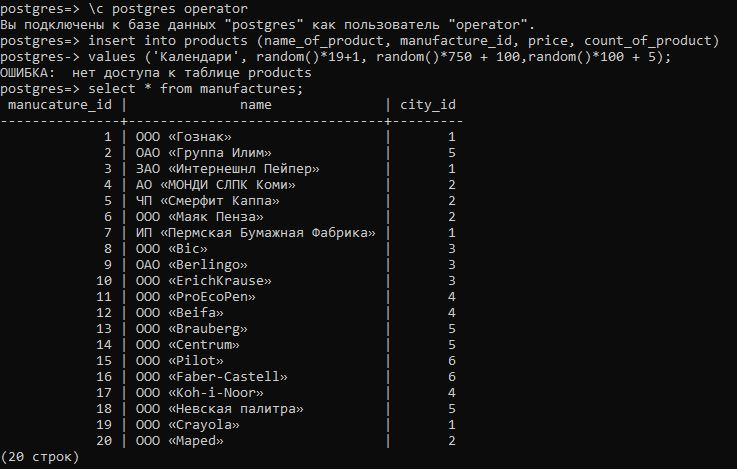
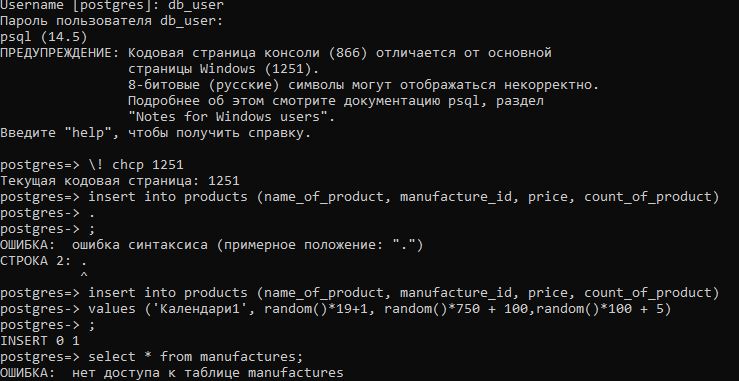
end;

$$ language plpgsql;



*Рис. 28. Процент полиграфической продукции для заданного возраста.*

# Проверка работы ролей



*Рис. 29. Проверка правильности работы ролей.*

Как можно видеть на рисунке 29, администратор (admin) имеет доступ ко всем операциям со всеми таблицами в базе данных. Оператор (operator) имеет доступ только к таблицам-справочникам: countries, clients, vendors, manufacturers. Пользователь (db\_user) имеет доступ к таблицам с данными о товарах (products) и продажах (sales). Справочники он может только просматривать.

# Анализ результатов и выводы

В ходе выполнения работы была построена база данных «Полиграфическая продукция». В базе данных присутствуют таблицы-справочники и таблицы, использующие справочники. В пояснительной записке представлена структура таблиц и связи между ними. Также были реализованы запросы согласно варианту в виде функций, их описание представлено в разделе «Руководство пользователя».

Для реализации базы данных были применены знания и навыки разработки БД, выполнения запросов, написания функций, полученные в ходе выполнения лабораторных работ по курсу и изучения документации PostgreSQL.

# Список использованных источников

1. PostgreSQL / [Электронный ресурс] // PostgreSQL : Документация : [сайт]. — URL: https://postgrespro.ru/docs/postgresql (дата обращения: 01.12.2022).
2. Работа с PostgreSQL / [Электронный ресурс] // Proglib : [сайт]. — URL: https://proglib.io/p/learn-postgresql (дата обращения: 01.12.2022).
3. 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: 01.11.2022).

# Листинг функций, используемых в запросах

/\*1)Для каждого вида полиграфической продукции указать сведения о ней

(наименование, издательство, цена, тираж, и т.п.)\*/

select \* from products\_info();

/\*2)Для каждого вида полиграфической продукции получить список, отсортированный:

в алфавитном порядке по наименованию, по тиражу, по стоимости\*/

select \* from products

order by name\_of\_product;

select \* from products

order by count\_of\_product;

select \* from products

order by price;

/\*3)Найти самую дорогую полиграфическую продукцию, самую дешевую, среднюю стоимость\*/

select min(price) as min\_price,

avg(price) as avg\_price,

max(price) as max\_price

from products;

/\*4)Найти всю полиграфическую продукцию, с ценой выше заданной\*/

select \* from products\_price\_more\_then(300);

/\*5)Найти всю полиграфическую продукцию, чей тираж попадает в заданные интервалы(ввод)\*/

select \* from products\_count\_between(20, 50);

/\*6)Для заданного издательства(выбор) найти всю полиграфическую продукцию\*/

select \* from products\_by\_manufactures('ООО «Маяк Пенза»');

/\*7)Найти всю полиграфическую продукцию, чья цена попадает в заданные интервалы и для заданного издательства\*/

select \* from products\_price\_between\_and\_manufactures(150, 900 , 'ЗАО «Интернешнл Пейпер»');

/\*Найти всю полиграфическую продукцию, чья цена попадает в заданные интервалы\*/

/\*select \* from products\_price\_between(200, 600);\*/

/\*8)Найти долю дешевой полиграфической продукции\*/

select products\_percentage\_price\_less\_then(500) || '%';

/\*9)Найти долю полиграфической продукции, проданной за определенный период\*/

select products\_percentage\_data\_between('2021-02-11'::date, '2023-11-11'::date) || '%';

/\*10)Найти всю полиграфическую продукцию, поступившую от заданного издателя,

чья стоимость выше заданной\*/

select \* from products\_price\_more\_then\_and\_manufactures(600, 'ООО «Pilot»');

/\*11)Найти всю полиграфическую продукцию, поступившую от заданного издательства, чья

стоимость больше чем средняя стоимость продукции, поступившей из заданного города\*/

select \* from products\_manufacture\_city('ООО «Невская палитра»', 'Лондон');

/\*12)Найти долю дорогой полиграфической продукции, поступившей от заданного производителя,

и в целом\*/

select \* from products\_percentage\_price\_more\_then\_and\_manufactures(600, 'ООО «Pilot»');

/\*13)Найти среднюю стоимость продукции, проданной за определенный промежуток времени\*/

select products\_avg\_data\_between('2020-02-11'::date, '2022-11-11'::date);

/\*14)Найти всю полиграфическую продукцию, чья стоимость выше, чем средняя стоимость

полиграфической продукции заданного издательства\*/

select \* from products\_price\_more\_then\_avg\_in\_manufactures('ООО «Pilot»');

/\*15)Найти всю полиграфическую продукцию, которую лучше всего раскупают\*/

select \* from best\_saled\_products();

/\*16)Найти долю полиграфической продукции для заданного возраста

от общего числа полиграфической продукции\*/

select products\_percentage\_age('для детей') || '%';

drop function products\_info;

create or replace function products\_info()

returns table

(

name\_of\_product varchar(30),

manufacture varchar(30),

vendor varchar(30),

price integer,

city varchar(30),

count\_of\_product integer

)

as

$$

begin

return query select

products.name\_of\_product,

m.name,

v.name,

products.price,

c.city,

products.count\_of\_product

from products

join manufactures m on m.manucature\_id = products.manufacture\_id

join cities c on c.city\_id = m.city\_id

join vendors v on products.vendor\_id = v.vendor\_id;

end;

$$ language plpgsql;

drop function products\_price\_more\_then;

create or replace function products\_price\_more\_then(a integer)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer,

age varchar(100)

)

as

$$

declare

\_a integer;

begin

select a into \_a;

return query select \*

from products

where products.price > \_a;

end;

$$ language plpgsql;

drop function products\_price\_between;

create or replace function products\_price\_between(a integer, b integer)

returns table

(

name\_of\_product varchar(30),

price integer,

count integer

)

as

$$

declare

\_a integer;

\_b integer;

begin

select a into \_a;

select b into \_b;

return query select

products.name\_of\_product,

products.price,

products.count\_of\_product

from products

where products.price between a and b;

end;

$$ language plpgsql;

create or replace function products\_price\_between\_and\_manufactures(a integer, b integer, name text)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer

)

as

$$

declare

\_a integer;

\_b integer;

\_name text;

begin

select a into \_a;

select b into \_b;

select name into \_name;

return query select

products.product\_id,

products.name\_of\_product,

products.manufacture\_id,

products.vendor\_id,

products.price,

products.count\_of\_product

from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

where products.price between a and b and m.name = \_name;

end;

$$ language plpgsql;

drop function products\_count\_between;

create or replace function products\_count\_between(a integer, b integer)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

count\_of\_product integer,

age varchar(100)

)

as

$$

declare

\_a integer;

\_b integer;

begin

select a into \_a;

select b into \_b;

return query select \*

from products

where products.count\_of\_product between a and b;

end;

$$ language plpgsql;

create or replace function products\_by\_manufactures(\_manufact text)

returns table(

name\_of\_product varchar(30),

price integer,

count\_of\_product integer

)

as

$$

begin

return query select

products.name\_of\_product,

products.price,

products.count\_of\_product

from products

JOIN manufactures m on products.manufacture\_id = m.manucature\_id

where m.name = \_manufact;

end;

$$ language plpgsql;

create or replace function products\_percentage\_price\_less\_then(a integer)

returns integer

as

$$

declare

\_cheap integer;

\_all integer;

begin

select count(name\_of\_product) into \_all from products;

select count(name\_of\_product) into \_cheap from products

where price < a;

return round((\_cheap::double precision / \_all::double precision) \* 100);

end;

$$ language plpgsql;

create or replace function products\_percentage\_data\_between(a date, b date)

returns integer

as

$$

declare

count integer;

\_all integer;

begin

select count(date\_of\_sale) into count from sales

where date\_of\_sale between a and b;

select count(date\_of\_sale) into \_all from sales;

return round((count::double precision / \_all::double precision) \* 100);

end;

$$ language plpgsql;

create or replace function products\_price\_more\_then\_and\_manufactures(a integer, name text)

returns table

(

name\_of\_product varchar(30),

manufacture varchar(30),

price integer,

count\_of\_product integer

)

as

$$

declare

\_a integer;

\_name text;

begin

select a into \_a;

select name into \_name;

return query select

products.name\_of\_product,

m.name,

products.price,

products.count\_of\_product

from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

where products.price > a and m.name = \_name;

end;

$$ language plpgsql;

drop function products\_manufacture\_city;

create or replace function products\_manufacture\_city(\_manufact text, \_city text)

returns table

(

name\_of\_product varchar(30),

manufuct varchar(30),

price integer,

avg\_price integer,

count\_of\_product integer

)

as

$$

declare

avg\_price integer;

begin

select avg(products.price) into avg\_price from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

JOIN cities as c ON m.city\_id = c.city\_id

where c.city = \_city;

return query select

products.name\_of\_product,

m.name,

products.price,

avg\_price,

products.count\_of\_product

from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

JOIN cities as c ON m.city\_id = c.city\_id

where m.name = \_manufact and products.price >avg\_price;

end;

$$ language plpgsql;

create or replace function products\_percentage\_price\_more\_then\_and\_manufactures(\_price integer, \_manufact text)

returns table(

expensive integer,

expensive\_all integer

)

as

$$

declare

\_expensive\_manufact integer;

\_expensive integer;

\_all integer;

begin

select count(products.name\_of\_product) into \_expensive\_manufact from products

JOIN manufactures m on products.manufacture\_id = m.manucature\_id

where products.price > \_price and m.name = \_manufact;

select count(products.name\_of\_product) into \_expensive from products

where products.price > \_price;

select count(products.name\_of\_product) into \_all from products;

return query select round((\_expensive\_manufact::double precision/\_all::double precision)\*100)::integer,

round((\_expensive::double precision/\_all::double precision)\*100)::integer;

end

$$ language plpgsql;

create or replace function products\_avg\_data\_between(date1 date, date2 date)

returns integer

as

$$

declare

count integer;

begin

select avg(products.price) into count from products

join sales s on products.product\_id = s.product\_id

where date\_of\_sale between date1 and date2;

return count;

end;

$$ language plpgsql;

create or replace function products\_price\_more\_then\_avg\_in\_manufactures(manufact text)

returns table

(

product\_id integer,

name\_of\_product varchar(30),

manufacture\_id integer,

vendor\_id integer,

price integer,

avg\_prc integer

)

as

$$

declare

avg\_price integer;

begin

select avg(products.price) into avg\_price from products

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

where m.name = manufact;

return query select

products.product\_id,

products.name\_of\_product,

products.manufacture\_id,

products.vendor\_id,

products.price,

avg\_price

from products

where products.price > avg\_price;

end;

$$ language plpgsql;

create or replace function best\_saled\_products()

returns table

(

product varchar(30),

manufacture varchar(30),

vendor varchar(30),

price integer,

count\_of\_sales integer

)

as

$$

begin

return query select

products.name\_of\_product,

m.name,

v.name,

products.price,

count(products.name\_of\_product)::integer as count\_of\_products

from products

JOIN sales s on products.product\_id = s.product\_id

JOIN manufactures m on m.manucature\_id = products.manufacture\_id

JOIN vendors v on products.vendor\_id = v.vendor\_id

group by s.product\_id,

products.product\_id,

m.manucature\_id,

v.vendor\_id

order by count\_of\_products desc

limit 4;

end;

$$ language plpgsql;

create or replace function products\_percentage\_age(\_age text)

returns integer

as

$$

declare

count integer;

\_all integer;

begin

select count(name\_of\_product) into count from products

where products.age = \_age;

select count(name\_of\_product) into \_all from products;

return round((count::double precision / \_all::double precision) \* 100);

end;

$$ language plpgsql;

***Листинг создания БД***

drop role if exists operator;

drop role if exists db\_user;

drop role if exists admin;

create table client

(

client\_id serial primary key,

surname varchar(30),

age integer

);

create table products

(

product\_id serial primary key,

name\_of\_product varchar(30),

manufacture\_id integer references manufactures (manucature\_id),

vendor\_id integer references vendors (vendor\_id),

price integer,

count\_of\_product integer,

age varchar(80)

);

create table sales

(

sales\_id serial primary key,

product\_id integer references products (product\_id),

client\_id integer references client (client\_id),

date\_of\_sale date

);

create table manufactures

(

manucature\_id serial primary key,

name varchar(30),

city\_id integer references cities (city\_id)

);

create table cities

(

city\_id serial primary key,

city varchar(30)

);

create table vendors

(

vendor\_id serial primary key,

name varchar(30)

);

create role operator with login password 'operator';

create role db\_user with login password 'user';

create role admin with login password 'admin';

grant usage on schema public to operator;

grant usage on schema public to db\_user;

grant usage on schema public to admin;

grant all on schema public to admin;

grant all on schema public to operator;

grant all on schema public to db\_user;

grant select, usage on all sequences in schema public to operator;

grant select, usage on all sequences in schema public to db\_user;

grant all privileges on all sequences in schema public to admin;

grant insert, select, update, delete on cities, manufactures to operator;

grant insert, select, update, delete on products, sales to db\_user;

grant all privileges on all tables in schema public to admin;

insert into cities (city)

values ('Благовещенск'),

('Чебоксары'),

('Лондон'),

('Абергел'),

('Далян'),

('Гонконг'),

('Берлин'),

('Мюнхен'),

('Ташкент'),

('Самарканд'),

('Сан-Паулу'),

('Рио-де-Жанейро');

insert into manufactures (name, city\_id)

values ('ООО «Гознак»', 1),

('ОАО «Группа Илим»', 5),

('ЗАО «Интернешнл Пейпер»', 1),

('АО «МОНДИ СЛПК Коми»', 2),

('ЧП «Смерфит Каппа»', 2),

('ООО «Маяк Пенза»', 2),

('ИП «Пермская Бумажная Фабрика»', 1),

('ООО «Bic»', 3),

('ОАО «Berlingo»', 3),

('ООО «ErichKrause»', 3),

('ООО «ProEcoPen»', 4),

('ООО «Beifa»', 4),

('ООО «Brauberg»', 5),

('ООО «Centrum»', 5),

('ООО «Pilot»', 6),

('ООО «Faber-Castell»', 6),

('ООО «Koh-i-Noor»', 4),

('ООО «Невская палитра»', 5),

('ООО «Crayola»', 1),

('ООО «Maped»', 2);

insert into client (surname, age)

values ('Иванов', random()\*40+8),

('Смирнов', random()\*40+8),

('Кузнецов', random()\*40+8),

('Попов', random()\*40+8),

('Васильев', random()\*40+8),

('Петров', random()\*40+8),

('Соколов', random()\*40+8),

('Михайлов', random()\*40+8),

('Новиков', random()\*40+8),

('Фёдоров', random()\*40+8),

('Морозов', random()\*40+8),

('Волков', random()\*40+8),

('Алексеев', random()\*40+8),

('Лебедев', random()\*40+8),

('Семёнов', random()\*40+8),

('Егоров', random()\*40+8);

insert into products (name\_of\_product, manufacture\_id, price, count\_of\_product)

values ('Календари', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Блокноты', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Листовки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Плакаты', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Буклеты', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Брашюры', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Журналы', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Книги', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Грамоты', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Кубарики', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Папки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Упаковки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Визитки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Открытки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Бланки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Билеты', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Планинги', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Каталоги', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Наклейки', random()\*19+1, random()\*750 + 100,random()\*100 + 5),

('Атласы', random()\*19+1, random()\*750 + 100,random()\*100 + 5);

insert into vendors (name)

values ('Лазерпринт'),

('Атмосфера'),

('Mix'),

('Лемур'),

('Советская Сибирь'),

('Центральный'),

('Кант'),

('Школьник'),

('Авантия-принт'),

('Принта'),

('iPrint'),

('АкадемПостер');

insert into sales (client\_id, product\_id, date\_of\_sale)

values (random()\*15+1, random()\*19+1, '2022-06-01'),

(random()\*15+1, random()\*19+1, '2022-06-04'),

(random()\*15+1, random()\*19+1, '2022-06-10'),

(random()\*15+1, random()\*19+1, '2022-06-15'),

(random()\*15+1, random()\*19+1, '2022-07-25'),

(random()\*15+1, random()\*19+1, '2022-07-21'),

(random()\*15+1, random()\*19+1, '2021-02-15'),

(random()\*15+1, random()\*19+1, '2021-01-26'),

(random()\*15+1, random()\*19+1, '2021-08-12'),

(random()\*15+1, random()\*19+1, '2021-09-20'),

(random()\*15+1, random()\*19+1, '2021-10-24'),

(random()\*15+1, random()\*19+1, '2021-12-01'),

(random()\*15+1, random()\*19+1, '2021-11-05'),

(random()\*15+1, random()\*19+1, '2021-02-08'),

(random()\*15+1, random()\*19+1, '2021-01-14'),

(random()\*15+1, random()\*19+1, '2021-03-11'),

(random()\*15+1, random()\*19+1, '2021-04-21'),

(random()\*15+1, random()\*19+1, '2021-05-21'),

(random()\*15+1, random()\*19+1, '2021-02-26'),

(random()\*15+1, random()\*19+1, '2021-01-15'),

(random()\*15+1, random()\*19+1, '2021-07-14'),

(random()\*15+1, random()\*19+1, '2021-09-21'),

(random()\*15+1, random()\*19+1, '2021-10-16'),

(random()\*15+1, random()\*19+1, '2022-11-19'),

(random()\*15+1, random()\*19+1, '2022-10-20'),

(random()\*15+1, random()\*19+1, '2020-11-24'),

(random()\*15+1, random()\*19+1, '2021-11-28');