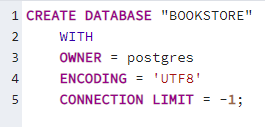
Дмитрий Варгин,

Ксения Вотинова

1 вариант

**Задание 1**

**Создаем базу данных:**



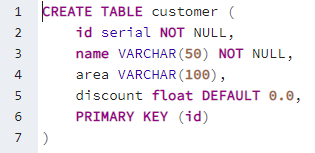
**Таблицы:**

NOT NULL было использовано тогда, когда без этого признака была бы бесполезна вся строка в таблице и значение NULL нельзя заменить на другое однозначно

Для второго условия использовался DEFAULT

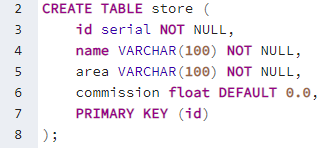
**Таблица покупателей:**

CREATE TABLE customer (id serial NOT NULL, name VARCHAR(50) NOT NULL, area VARCHAR(100), discount float DEFAULT 0.0, PRIMARY KEY (id) );



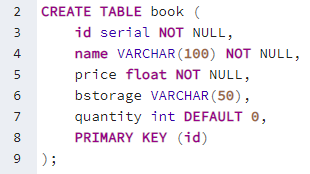
**Таблица магазинов:**

CREATE TABLE store (id serial NOT NULL, name VARCHAR(100) NOT NULL, area VARCHAR(100) NOT NULL, commission float DEFAULT 0.0, PRIMARY KEY (id) );



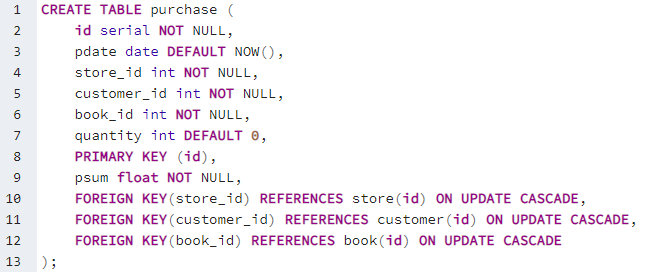
**Таблица книг:**

CREATE TABLE book ( id serial NOT NULL, name VARCHAR(100) NOT NULL, price float NOT NULL, bstorage VARCHAR(50), quantity int DEFAULT 0, PRIMARY KEY (id) );



**Таблица покупок:**

CREATE TABLE purchase (id serial NOT NULL, pdate date DEFAULT NOW(), store\_id int NOT NULL, customer\_id int NOT NULL, book\_id int NOT NULL, quantity int DEFAULT 0, PRIMARY KEY (id), psum float NOT NULL, FOREIGN KEY(store\_id) REFERENCES store(id) ON UPDATE CASCADE, FOREIGN KEY(customer\_id) REFERENCES customer(id) ON UPDATE CASCADE, FOREIGN KEY(book\_id) REFERENCES book(id) ON UPDATE CASCADE);



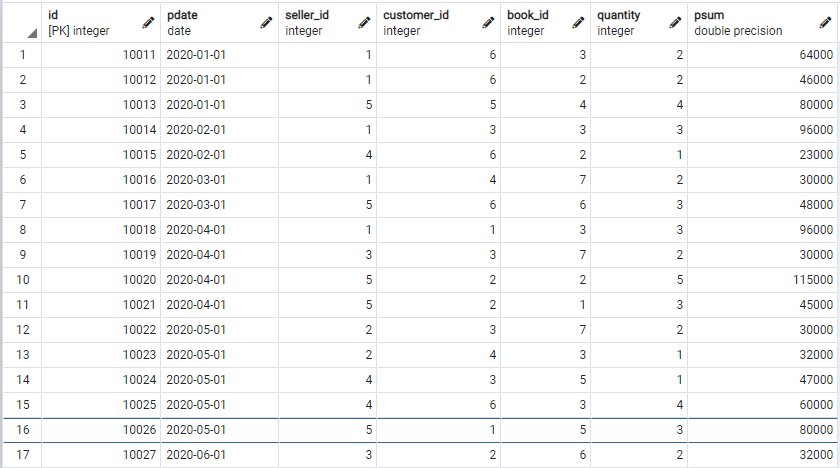
**Задание 2**

С помощью скрипта на питоне(https://github.com/Dmitry-Vargin-HSE-works/SQLab1/blob/master/main.py) перевели таблицу в SQL запросы.

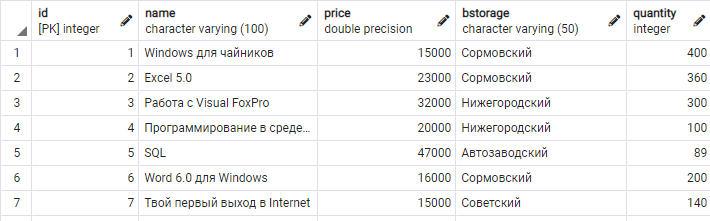
**Задание 3**

**Используя оператор SELECT создать запрос для вывода всех строк каждой таблицы. Проверить правильность ввода.**

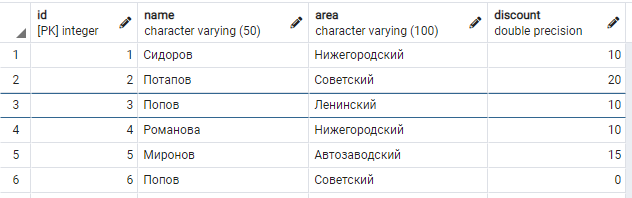
SELECT \* FROM purchase;



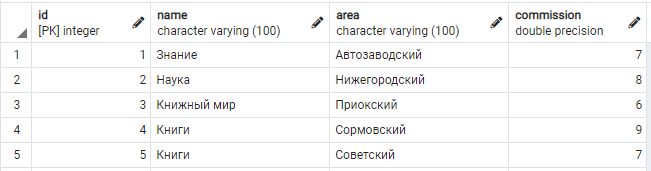
SELECT \* FROM book;



SELECT \* FROM customer;



SELECT \* FROM store;



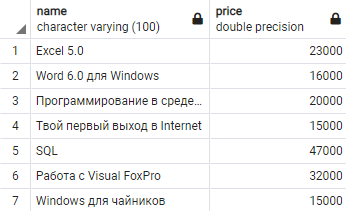
Все верно.

**Задание 4**

Создать запросы для вывода:

1. всех различных названий и стоимостей книг;

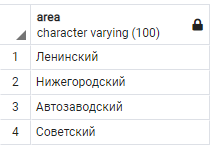
SELECT DISTINCT name, price FROM book;



Так как все имена и стоимости уникальны получилась такая же таблица как если бы мы не использовали DISTINCT.

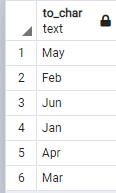
1. всех различных районов, в которых проживают покупатели;

SELECT DISTINCT area FROM customer;



1. всех различных месяцев, когда производились покупки.

SELECT DISTINCT to\_char(pdate, 'Mon') FROM purchase;



**Задание 6**

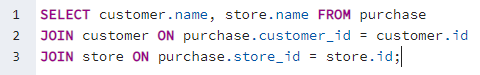
Для каждой покупки вывести следующие данные:

1. фамилию покупателя и название магазина, где производилась покупка;

SELECT customer.name, store.name FROM purchase

JOIN customer ON purchase.customer\_id = customer.id

JOIN store ON purchase.store\_id = store.id;





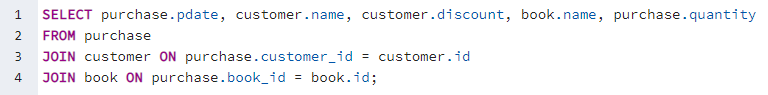
1. дату, фамилию покупателя, скидку, название и количество купленных книг.

SELECT purchase.pdate, customer.name, customer.discount, book.name, purchase.quantity

FROM purchase

JOIN customer ON purchase.customer\_id = customer.id

JOIN book ON purchase.book\_id = book.id;

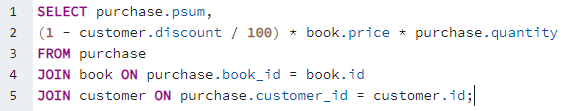


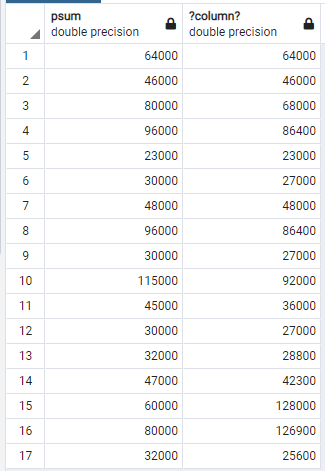


**Задание 8**

Создать запрос для модификации всех значений столбца с суммарной величиной покупки, чтобы он содержал истинную сумму, оплачиваемую покупателем ( с учетом скидки). Вывести новые значения.

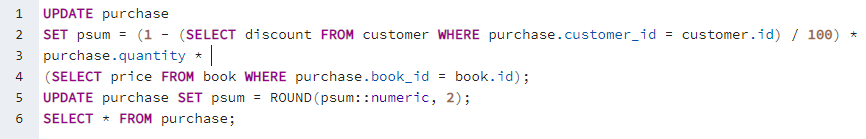
SELECT purchase.psum, (1 - customer.discount / 100) \* book.price \* purchase.quantity FROM purchase JOIN book ON purchase.book\_id = book.id JOIN customer ON purchase.customer\_id = customer.id;

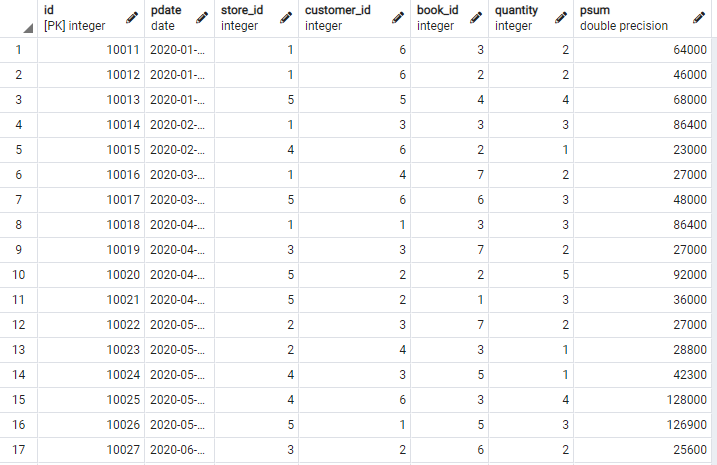




Можно заметить, что бОльшая часть значений изменилась

UPDATE purchase SET psum = (1 - (SELECT discount FROM customer WHERE purchase.customer\_id = customer.id) / 100) \* purchase.quantity \* (SELECT price FROM book WHERE purchase.book\_id = book.id); UPDATE purchase SET psum = ROUND(psum::numeric, 2); SELECT \* FROM purchase;



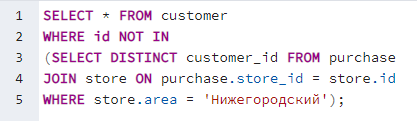


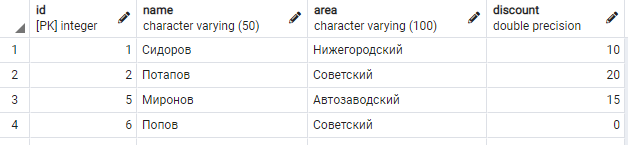
**Задание 10**

**Используя операцию IN (NOT IN) реализовать следующие запросы:**

1. **найти покупателей, которые не покупали книг в магазинах Нижегородского района в июне;**

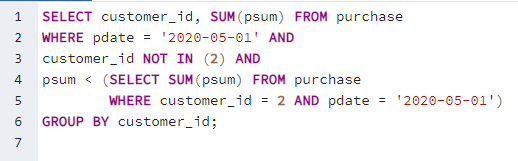
SELECT \* FROM customer WHERE id NOT IN (SELECT DISTINCT customer\_id FROM purchase JOIN store ON purchase.store\_id = store.id WHERE store.area = 'Нижегородский');





1. **найти покупателей, покупавших книги в мае на сумму, меньшую чем купил Потапов в том же месяце;**

SELECT customer\_id, SUM(psum) FROM purchase WHERE pdate = '2020-05-01' AND customer\_id NOT IN (2) AND psum < (SELECT SUM(psum) FROM purchase WHERE customer\_id = 2 AND pdate = '2020-05-01') GROUP BY customer\_id;



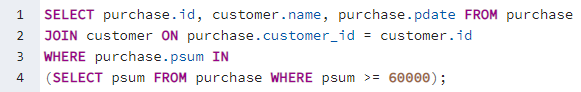
Запрос ничего не выводит так как в таблице нет информации о покупках Потапова в мае

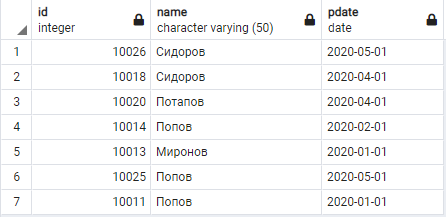
1. **реализовать запросы заданий 7.а, 7.с.**

**Определить**

1. **номер заказа, фамилию покупателя и дату для покупок в которых было продано книг на сумму не меньшую чем 60000 руб.**

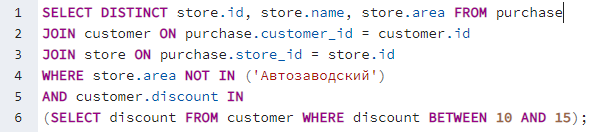
SELECT purchase.id, customer.name, purchase.pdate FROM purchase JOIN customer ON purchase.customer\_id = customer.id WHERE purchase.psum IN (SELECT psum FROM purchase WHERE psum >= 60000);

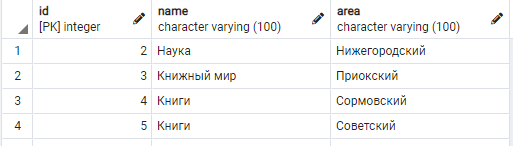




1. **магазины, расположенные в любом районе, кроме Автозаводского, где покупали книги те, у кого скидка от 10 до 15 %;**

SELECT DISTINCT store.id, store.name, store.area FROM purchase JOIN customer ON purchase.customer\_id = customer.id JOIN store ON purchase.store\_id = store.id WHERE store.area NOT IN ('Автозаводский') AND customer.discount IN (SELECT discount FROM customer WHERE discount BETWEEN 10 AND 15);

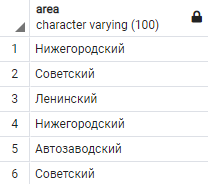




**Задание 12**

**Используя операцию UNION получить районы проживания покупателей и районы складирования книг.**

SELECT area FROM customer UNION SELECT bstorage FROM book;

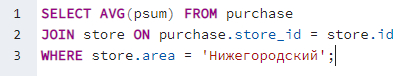
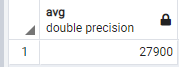


**Задание 14**

**Реализовать запросы с использованием агрегатных функций:**

1. **получить среднюю стоимость покупок, сделанных в магазинах Нижегородского района;**

SELECT AVG(psum) FROM purchase JOIN store ON purchase.store\_id = store.id WHERE store.area = 'Нижегородский';

1. **найти количество покупателей, покупавших книги в магазине “Наука”;**

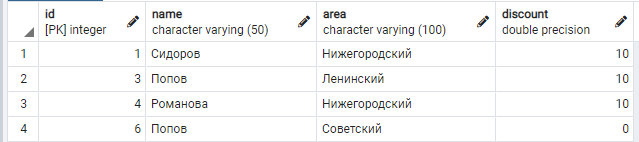
SELECT COUNT(DISTINCT purchase.customer\_id) FROM purchase JOIN store ON purchase.store\_id = store.id WHERE store.name = 'Наука';

1. **найти покупателей имеющих скидку ниже средней;**

SELECT \* FROM customer WHERE discount < (SELECT AVG(discount) FROM customer);





1. **определить магазины, в которых покупало книги больше покупателей чем в магазине “Наука”.**

SELECT store.id, store.name, tabq.num FROM (SELECT store\_id, COUNT(DISTINCT customer\_id) as num FROM purchase GROUP BY store\_id) as tabq JOIN store ON tabq.store\_id = store.id WHERE tabq.num > (SELECT COUNT(DISTINCT customer\_id) FROM purchase WHERE store\_id = 2);

