# Лабораторная работа 1

# Создание связи один к одному:

CREATE TABLE Man (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR

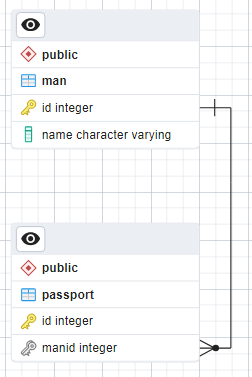
);

CREATE TABLE Passport (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

ManId int REFERENCES Man(id) UNIQUE

);



Заполним таблицы данными:

INSERT INTO Man (name) VALUES (‘Oleg’);

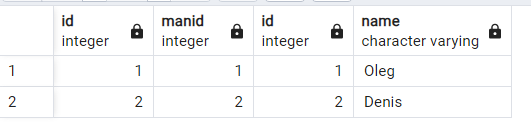
INSERT INTO Man (name) VALUES (‘Denis’);

INSERT INTO Passport (ManId) VALUES (1);

INSERT INTO Passport (ManId) VALUES (2);

Выводим результат на экран:

SELECT \* FROM Passport JOIN Man on Passport.ManId = Man.id;



# Создание связи один ко многим:

CREATE TABLE Street (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR

);

CREATE TABLE Home (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

number int,

StreetId int REFERENCES Street(id)

);

INSERT INTO Street (name) VALUES ('Дубосековская');

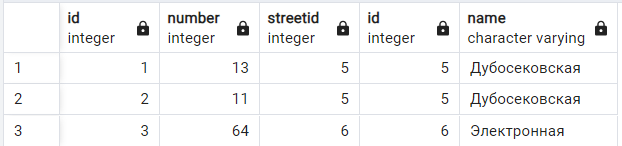
INSERT INTO Street (name) VALUES ('Электронная');

INSERT INTO Home (number) VALUES (13, 1);

INSERT INTO Home (number) VALUES (11 , 1);

INSERT INTO Home (number) VALUES (64 , 2);

SELECT \* FROM HOME JOIN Street ON Home.StreetId = Street.Id;



# Создадим связь многие ко многим:

CREATE TABLE Company (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

name VARCHAR

);

CREATE TABLE Person (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

username VARCHAR

);

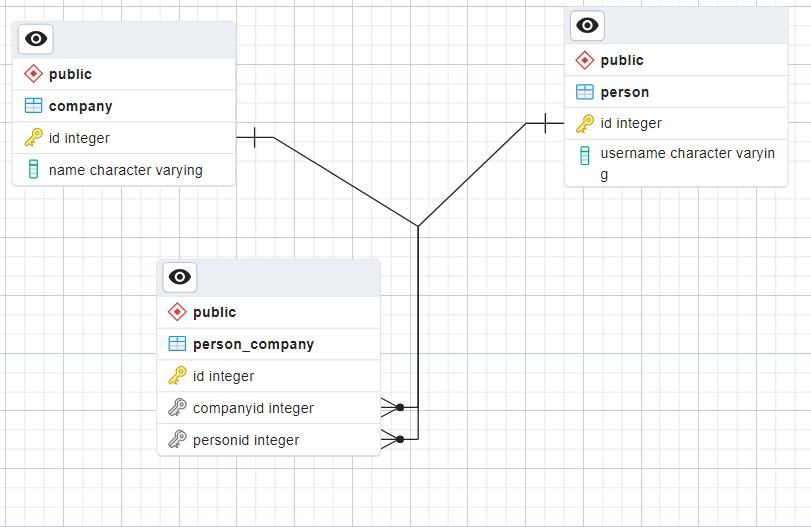
CREATE TABLE Person\_Company (

id int GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,

CompanyId int REFERENCES Company(id),

PersonId int REFERENCES Person(id)

);



# Теперь заполним таблицу:

INSERT INTO Company (name) VALUES ('Amazon');

INSERT INTO Company (name) VALUES (‘Tesla’);

INSERT INTO Person (username) VALUES ('Nikita');

INSERT INTO Person (username) VALUES ('Alecsandr');

INSERT INTO Person\_Company (CompanyId, PersonId ) VALUES (1, 2);

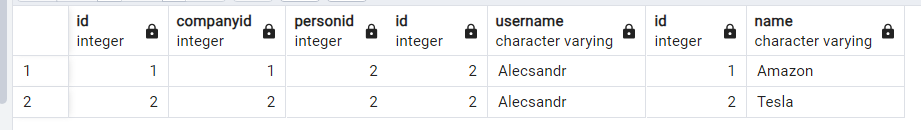
INSERT INTO Person\_Company (CompanyId, PersonId ) VALUES (2, 2);

Выводим результат:

SELECT \* FROM Person\_Company

JOIN PERSON on person.id = Person\_Company.personId

JOIN Company on company.id = Person\_Company.companyId ;

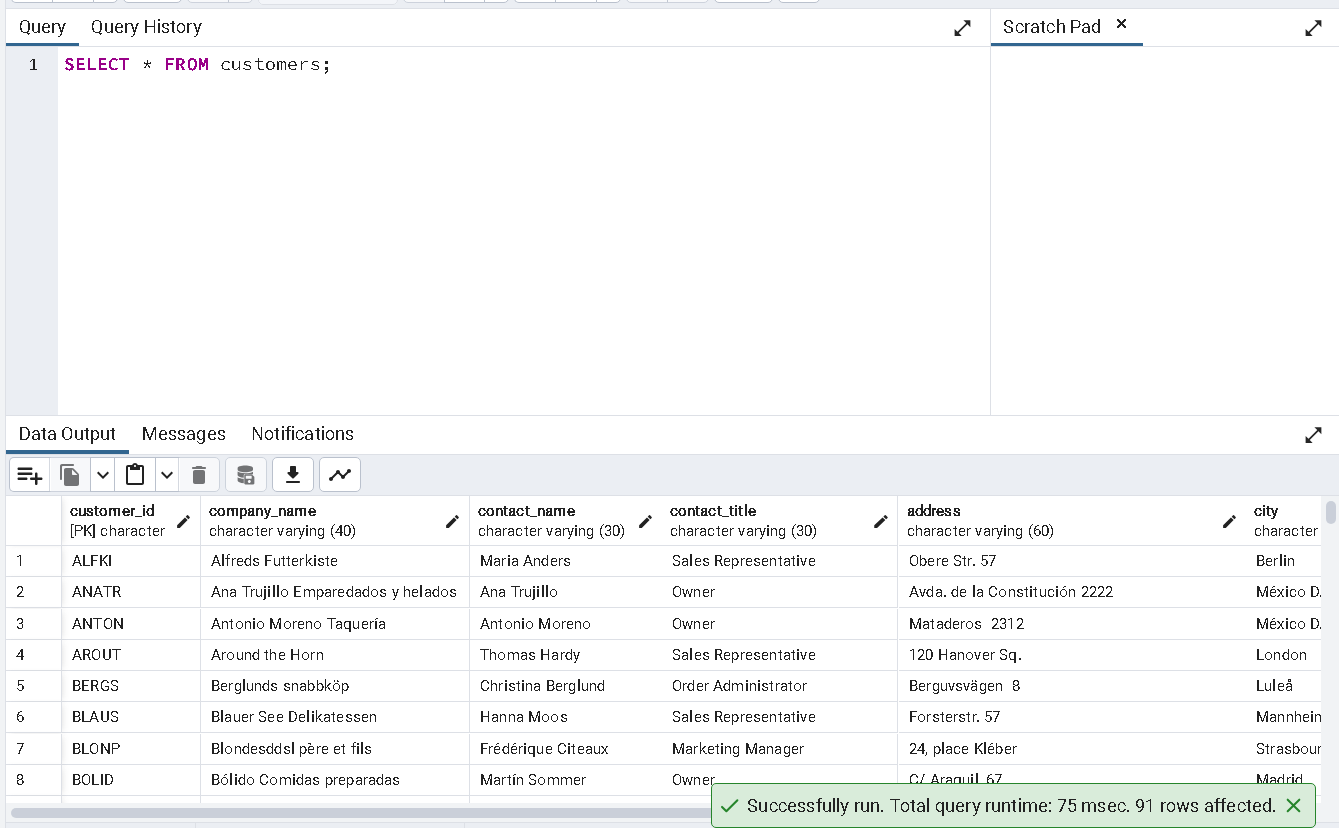


# Лабораторная работа 2

## **Задание 2.1**

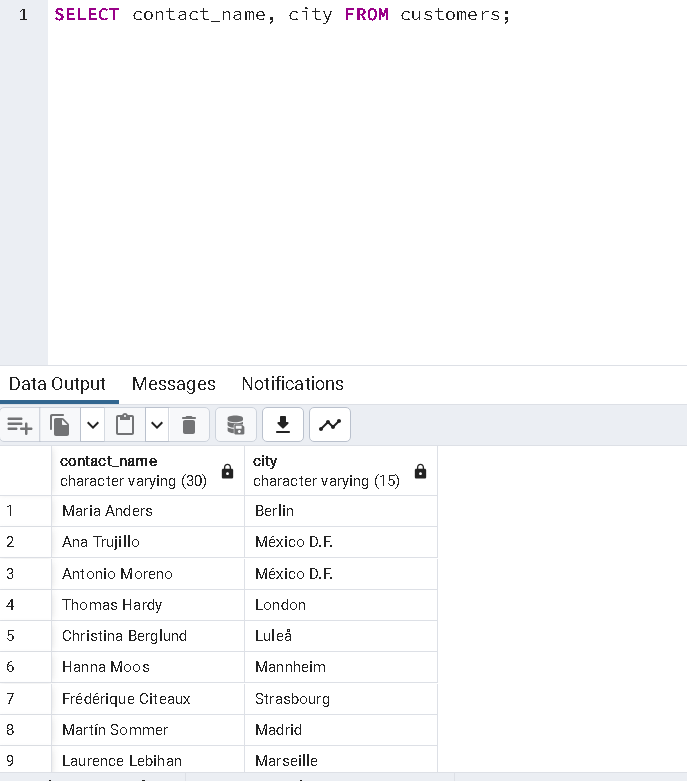
1. Выбрать все данные из таблицы customers

SELECT \* FROM customers;



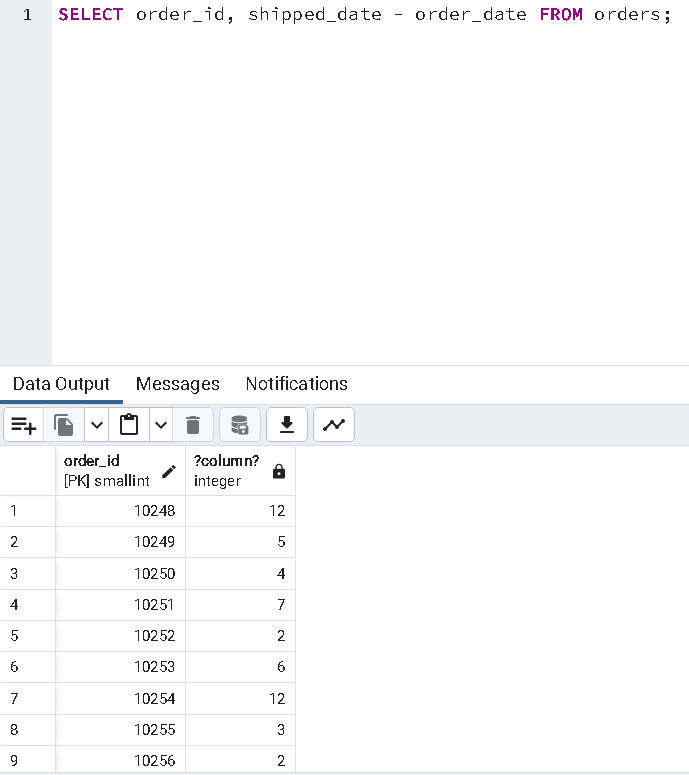
2. Выбрать все записи из таблицы customers, но только колонки "имя контакта" и "город"

SELECT contact\_name, city FROM customers;



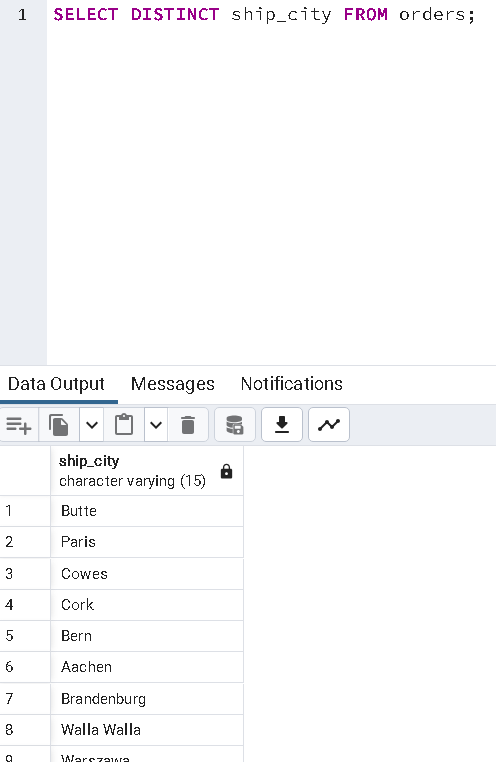
3. Выбрать все записи из таблицы orders, но взять две колонки: идентификатор заказа и колонку, значение в которой мы рассчитываем, как разницу между датой отгрузки и датой формирования заказа.

SELECT order\_id, shipped\_date - order\_date FROM orders;



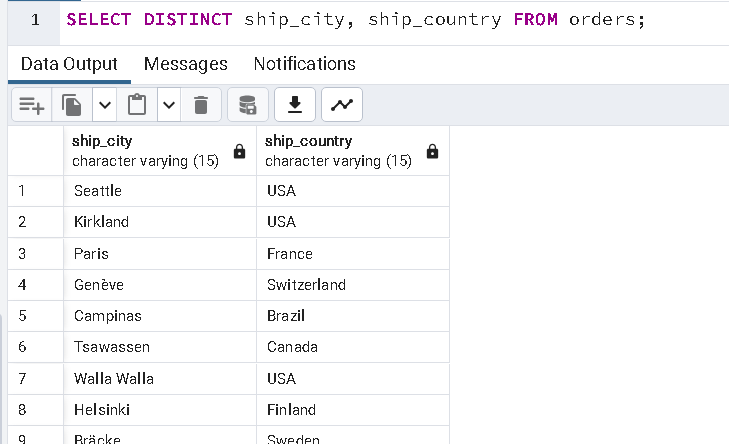
4. Выбрать все уникальные города, в которых "зарегистрированы" заказчики

SELECT DISTINCT ship\_city FROM orders;



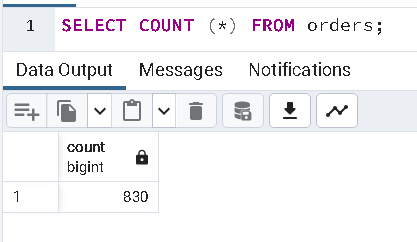
5. Выбрать все уникальные сочетания городов и стран, в которых "зарегистрированы" заказчики

SELECT DISTINCT ship\_city, ship\_country FROM orders;



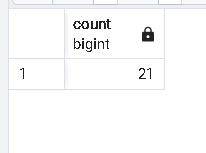
6. Посчитать кол-во заказчиков

SELECT COUNT (\*) FROM orders;



7. Посчитать кол-во уникальных стран, в которых "зарегистрированы" заказчики

SELECT COUNT (DISTINCT ship\_country) FROM orders;



## **Задание 2.2.**

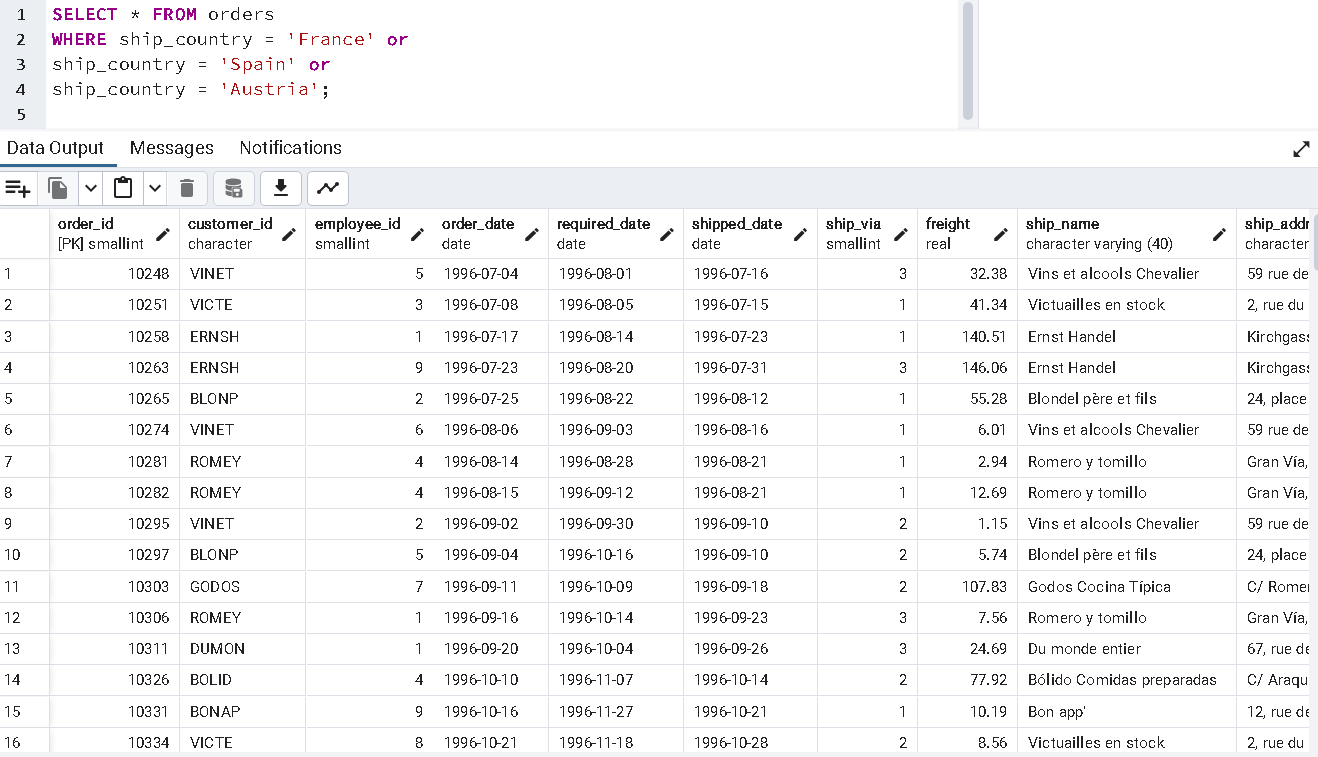
1. Выбрать все заказы из стран France, Austria, Spain

SELECT \* FROM orders

WHERE ship\_country = 'France' or

ship\_country = 'Spain' or

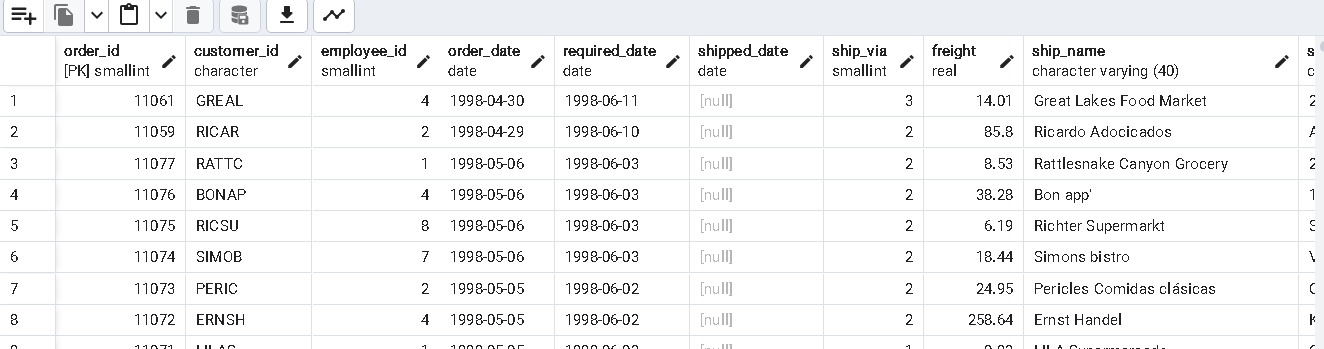
ship\_country = 'Austria';



1. Выбрать все заказы, отсортировать по required\_date (по убыванию) и отсортировать по дате отгрузке (по возрастанию)

SELECT \* FROM orders

ORDER BY required\_date DESC, shipped\_date ASC;



1. Выбрать минимальное кол-во единиц товара среди тех продуктов, которых в продаже более 30 единиц.

SELECT MIN (units\_in\_stock) FROM products WHERE units\_in\_stock > 30;



1. Выбрать максимальное кол-во единиц товара среди тех продуктов, которых в продаже более 30 единиц.

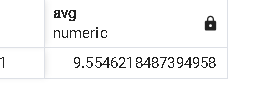
SELECT MAX (units\_in\_stock) FROM products WHERE units\_in\_stock > 30;



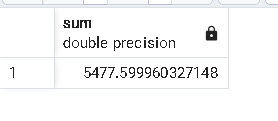
1. Найти среднее значение дней, уходящих на доставку с даты формирования заказа в USA

SELECT AVG (shipped\_date - order\_date)

FROM orders WHERE ship\_country = 'USA';



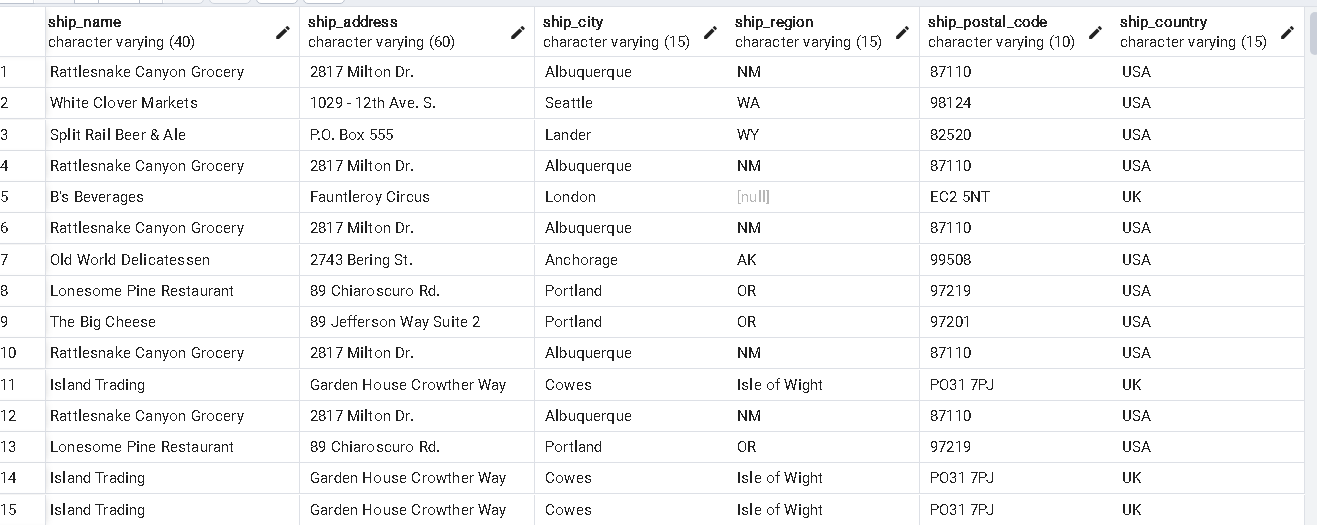
1. Найти сумму, на которую имеется товаров (кол-во \* цену) причём таких, которые планируется продавать и в будущем (см. на поле discontinued)

SELECT SUM (unit\_price\*units\_in\_stock) FROM products WHERE discontinued <> 0;  
  


## **Задание 2.3**

1. Выбрать все записи заказов, в которых наименование страны отгрузки начинается с 'U'

SELECT \* FROM orders WHERE ship\_country like 'U%';



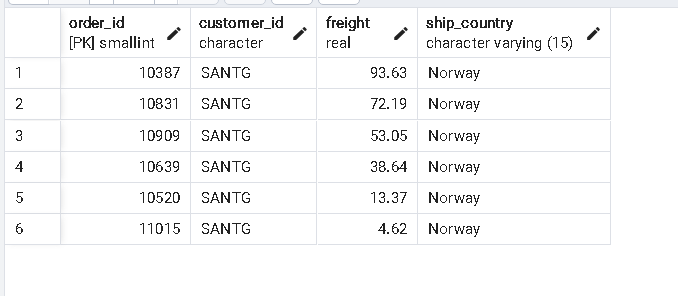
1. Выбрать записи заказов (включить колонки идентификатора заказа, идентификатора заказчика, веса и страны отгрузки), которые должны быть отгружены в страны имя которых начинается с 'N', отсортировать по весу (по убыванию) и вывести только первые 10 записей.

SELECT order\_id, customer\_id, freight, ship\_country

FROM orders WHERE ship\_country like 'N%'

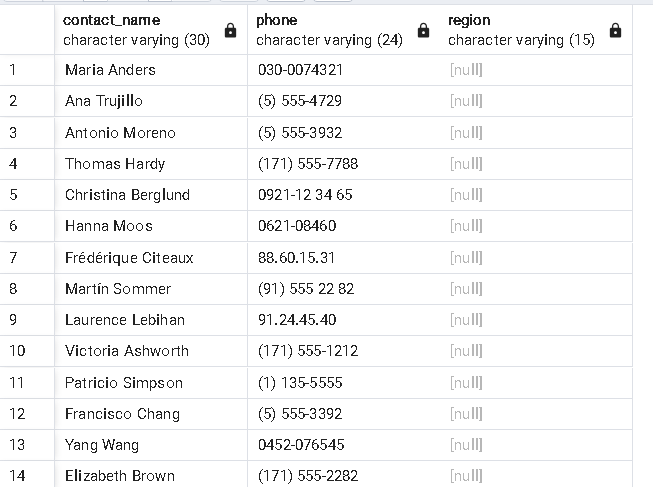
ORDER BY freight desc

limit 10;



1. Выбрать записи работников (включить колонки имени, фамилии, телефона, региона) в которых регион неизвестен

SELECT contact\_name, phone, region FROM customers WHERE region is NULL;



1. Подсчитать кол-во заказчиков регион которых известен

SELECT COUNT (\*) FROM customers WHERE region is NULL;



1. Подсчитать кол-во поставщиков в каждой из стран и отсортировать результаты группировки по убыванию кол-ва

SELECT country, COUNT (\*) AS suppliers\_count

FROM suppliers

GROUP BY country

ORDER BY suppliers\_count DESC;



1. Подсчитать суммарный вес заказов (в которых известен регион) по странам, затем отфильтровать по суммарному весу (вывести только те записи, где суммарный вес больше 2750) и отсортировать по убыванию суммарного веса.

SELECT ship\_country, SUM (freight) AS totalWeight

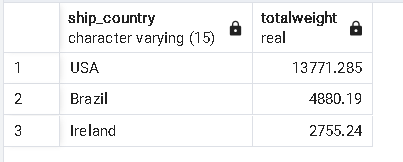
FROM orders

where ship\_region is not NULL

group by ship\_country

having SUM (freight) > 2750

ORDER BY SUM (freight) desc;



1. Выбрать все уникальные страны заказчиков и поставщиков и отсортировать страны по возрастанию

SELECT country FROM (

SELECT country AS country FROM customers

UNION

SELECT country AS country FROM suppliers

) AS countries

ORDER BY country ASC;



1. Выбрать такие страны, в которых "зарегистрированы" одновременно и заказчики, и поставщики, и работники.

SELECT distinct customers.country

FROM customers

JOIN suppliers ON customers.country = suppliers.country

JOIN employees ON customers.country = employees.country;



9. Выбрать такие страны в которых "зарегистрированы" одновременно заказчики и поставщики, но при этом в них не "зарегистрированы" работники.

SELECT distinct customers.country

FROM customers

JOIN suppliers ON customers.country = suppliers.country

JOIN employees ON customers.country <> employees.country;



# Лабораторная работа 3

## **Задание 1**

Найти заказчиков - customers и обслуживающих их заказы сотрудников - employee таких, что и заказчики, и сотрудники из города London, а доставка идёт компанией Speedy Express. Вывести компанию заказчика и ФИО сотрудника.

**SELECT customers.company\_name, employees.first\_name, employees.last\_name**

**FROM customers**

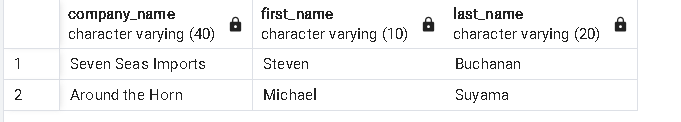
**INNER JOIN orders on customers.customer\_id = orders.customer\_id**

**INNER JOIN employees on employees.employee\_id = orders.employee\_id**

**WHERE customers.city = 'London'**

**AND employees.city = 'London'**

**AND orders.ship\_via = '1'**



## **Задание 2**

Найти активные (см. поле discontinued) продукты из категории Beverages и Seafood, которых в продаже менее 20 единиц. Вывести наименование продуктов, кол-во единиц в продаже, имя контакта поставщика и его телефонный номер.

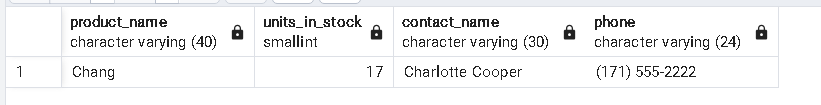
**SELECT products.product\_name, products.units\_in\_stock, suppliers.contact\_name, suppliers.phone**

**FROM products**

**INNER JOIN suppliers on products.supplier\_id = suppliers.supplier\_id**

**WHERE discontinued = '1'**

**AND (category\_id = '1' or category\_id = '8')**

**AND products.units\_in\_stock < 20**

## **Задание 3**

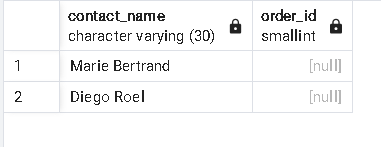
Найти заказчиков, не сделавших ни одного заказа. Вывести имя заказчика и order\_id.

**SELECT customers.contact\_name, orders.order\_id**

**FROM customers**

**LEFT JOIN orders on customers.customer\_id = orders.customer\_id**

**WHERE orders.order\_id is NULL;**



## **Задание 4**

Переписать предыдущий запрос, использовав симметричный вид джойна (подсказка: речь о LEFT и RIGHT).

**SELECT customers.contact\_name, orders.order\_id**

**FROM orders**

**RIGHT JOIN customers on customers.customer\_id = orders.customer\_id**

**Where orders.order\_id is NULL;**



# Лабораторная работа 4

## **Задание 1**

Вывести продукты количество которых в продаже меньше самого малого среднего количества продуктов в деталях заказов (группировка по product\_id). Результирующая таблица должна иметь колонки product\_name и units\_in\_stock.

**SELECT p.product\_name, p.units\_in\_stock**

**FROM products AS p**

**WHERE p.units\_in\_stock < (**

**SELECT MIN (avg\_qty)**

**FROM (**

**SELECT AVG(od.quantity) AS avg\_qty**

**FROM order\_details AS od**

**GROUP BY od.product\_id**

**)**

**);**



## **Задание 2**

Напишите запрос, который выводит общую сумму фрахтов заказов для компаний-заказчиков для заказов, стоимость фрахта которых больше или равна средней величине стоимости фрахта всех заказов, а также дата отгрузки заказа должна находится во второй половине июля 1996 года. Результирующая таблица должна иметь колонки customer\_id и freight\_sum, строки которой должны быть отсортированы по сумме фрахтов заказов.

**SELECT customer\_id, SUM(freight) AS freight\_sum**

**FROM orders**

**WHERE freight >= (**

**SELECT AVG(freight)**

**FROM orders**

**WHERE EXTRACT (MONTH FROM shipped\_date) = 7**

**AND EXTRACT(YEAR FROM shipped\_date) = 1996**

**AND EXTRACT(DAY FROM shipped\_date) >= 15**

**)**

**AND**

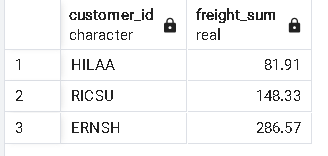
**EXTRACT(MONTH FROM shipped\_date) = 7**

**AND EXTRACT(YEAR FROM shipped\_date) = 1996**

**AND EXTRACT(DAY FROM shipped\_date) >= 15**

**GROUP BY customer\_id**

**ORDER BY freight\_sum;**



## **Задание 3**

Напишите запрос, который выводит 3 заказа с наибольшей стоимостью, которые были созданы после 1 сентября 1997 года включительно и были доставлены в страны Южной Америки. Общая стоимость рассчитывается как сумма стоимости деталей заказа с учетом дисконта. Результирующая таблица должна иметь колонки customer\_id, ship\_country и order\_price, строки которой должны быть отсортированы по стоимости заказа в обратном порядке.

**-- Задание 3**

**SELECT o.order\_id, o.ship\_country, (SELECT SUM(od.quantity \* od.unit\_price \* (1 - od.discount))**

**FROM order\_details AS od**

**WHERE od.order\_id = o.order\_id) as total\_cost**

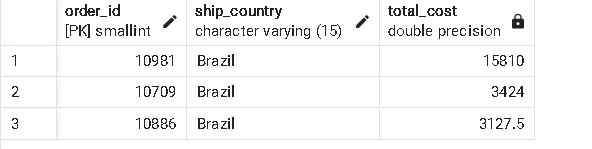
**FROM orders o**

**WHERE o.order\_date >= '1997-09-01'**

**AND o.ship\_country IN ('Argentina', 'Bolivia', 'Brazil', 'Chile', 'Columbia', 'Ecuador', 'Guyana', 'Paraguay', 'Peru', 'Suriname', 'Uruguay', 'Venezuela')**

**ORDER BY total\_cost DESC**

**LIMIT 3;**



## **Задание 4**

Вывести все товары (уникальные названия продуктов), которых заказано ровно 10 единиц (конечно же, это можно решить и без подзапроса).

**SELECT (SELECT product\_name**

**FROM products p**

**WHERE p.product\_id = od.product\_id) as name**

**FROM order\_details od**

**WHERE od.quantity = 10;**



# Лабораторная 5

## **Часть 1**

## **Задание 1**

Создать таблицу teacher с полями teacher\_id serial, first\_name varchar, last\_name varchar, birthday date, phone varchar, title varchar

**CREATE TABLE teacher**

**(**

**teacher\_id serial,**

**first\_name varchar,**

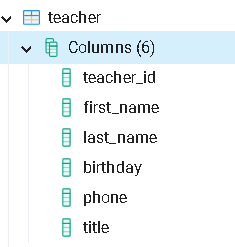
**last\_name varchar,**

**birthday date,**

**phone varchar,**

**title varchar**

**)**

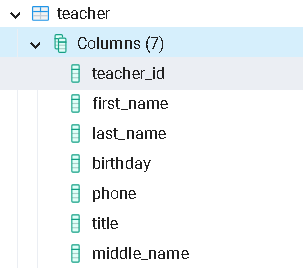


## **Задание 2**

Добавить в таблицу после создания колонку middle\_name varchar

**ALTER TABLE teacher**

**ADD COLUMN middle\_name varchar**



## **Задание 3**

Удалить колонку middle\_name

**ALTER TABLE teacher**

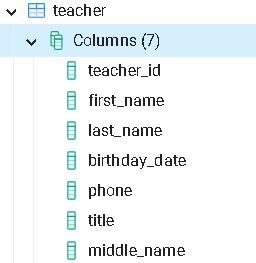
**DROP COLUMN middle\_name**

## **Задание 4**

Переименовать колонку birthday в birth\_date

**ALTER TABLE teacher**

**RENAME birthday TO birthday\_date**



## **Задание 5**

Изменить тип данных колонки phone на varchar(32)

**ALTER TABLE teacher**

**ALTER COLUMN phone SET DATA TYPE varchar(32)**

## **Задание 6**

Создать таблицу exam с полями exam\_id serial, exam\_name varchar(256), exam\_date date

**CREATE TABLE exam**

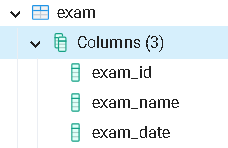
**(**

**exam\_id serial,**

**exam\_name varchar(256),**

**exam\_date date**

**);**



## **Задание 7**

Вставить три любых записи с автогенерацией идентификатора

**INSERT INTO exam (exam\_name, exam\_date)**

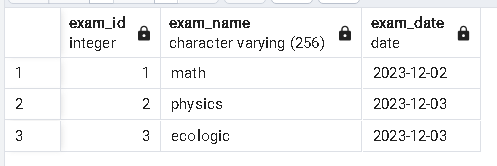
**VALUES**

**('math', '2023-12-2'),**

**('physics', '2023-12-3'),**

**('ecologic', '2023-12-3');**

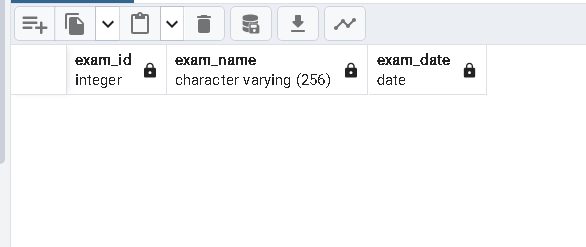
**SELECT \* FROM exam**



## **Задание 9**

Удалить все данные из таблицы со сбросом идентификатор в исходное состояние

**TRUNCATE TABLE exam RESTART IDENTITY**



## **Часть 2**

## **Задание 1**

Создать таблицу exam с полями:

- идентификатора экзамена - автоинкрементируемый, уникальный, запрещает NULL;- наименования экзамена- даты экзамена

**CREATE TABLE exam**

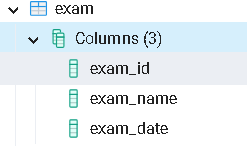
**(**

**exam\_id serial,**

**exam\_name varchar(256),**

**exam\_date date**

**);**



## **Задание 2**

Удалить ограничение уникальности с поля идентификатора

**ALTER TABLE exam**

**DROP CONSTRAINT exam\_exam\_id\_key;**

## **Задание 3**

Добавить ограничение первичного ключа на поле идентификатора

**ALTER TABLE exam**

**ADD PRIMARY KEY (exam\_id);**

## **Задание 4**

Создать таблицу person с полями

- идентификатора личности (простой int, первичный ключ)- имя- фамилия

**CREATE TABLE person**

**(**

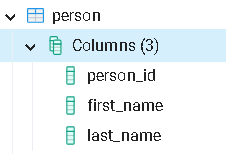
**person\_id integer NOT NULL,**

**first\_name varchar(64) NOT NULL,**

**last\_name varchar(64) NOT NULL,**

**CONSTRAINT pk\_person\_person\_id PRIMARY KEY(person\_id)**

**);**



## **Задание 5**

Создать таблицу паспорта с полями:

- идентификатора паспорта (простой int, первичный ключ)- серийный номер (простой int, запрещает NULL)- регистрация- ссылка на идентификатор личности (внешний ключ)

**CREATE TABLE passport**

**(**

**passport\_id int,**

**serial\_number int NOT NULL,**

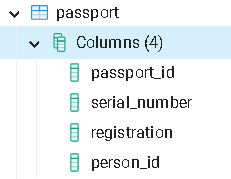
**registration text NOT NULL,**

**person\_id int NOT NULL,**

**CONSTRAINT pk\_passport\_passport\_id PRIMARY KEY(passport\_id),**

**CONSTRAINT fk\_passport\_person FOREIGN KEY (person\_id) REFERENCES person(person\_id)**

**);**



## **Задание 6**

Добавить колонку веса в таблицу book (создавали ранее) с ограничением, проверяющим вес (больше 0 но меньше 100)

## **Задание 7**

Убедиться в том, что ограничение на вес работает (попробуйте вставить невалидное значение)

**CREATE TABLE book**

**(**

**book\_id int,**

**nm int**

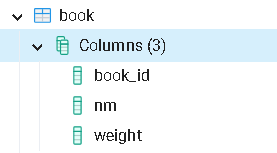
**);**

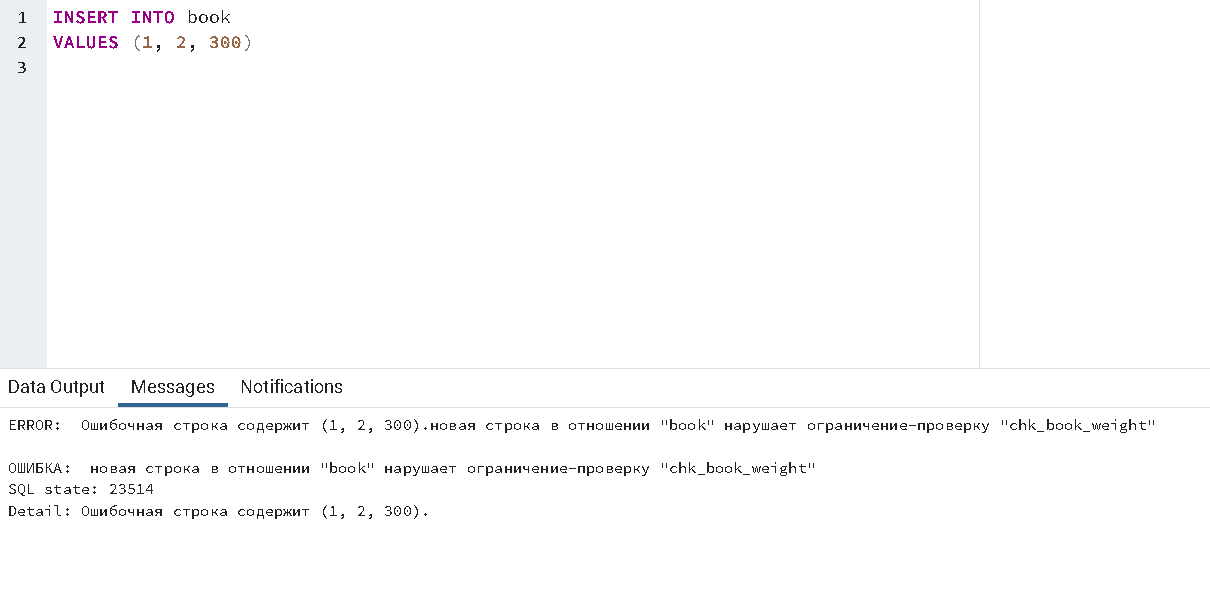
**ALTER TABLE book**

**ADD COLUMN weight decimal CONSTRAINT chk\_book\_weight CHECK (weight > 0 AND weight < 100)**

**INSERT INTO book**

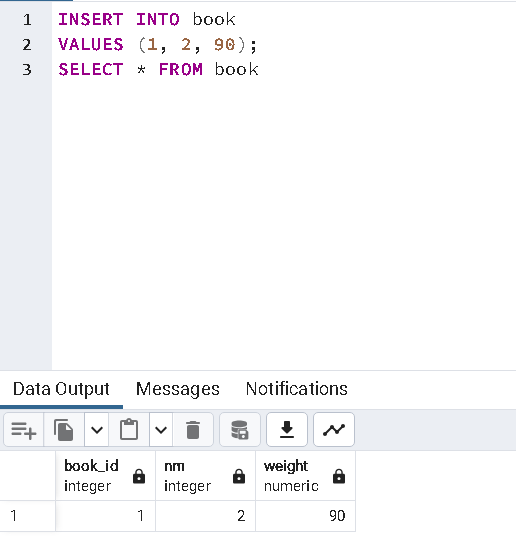
**VALUES (1, 2, 300)**





**INSERT INTO book**

**VALUES (1, 2, 90)**



## **Задание 8**

Создать таблицу student с полями:

- идентификатора (автоинкремент)- полное имя- курс (по умолчанию 1)

**CREATE TABLE student**

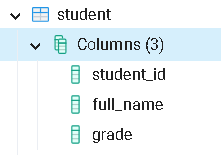
**(**

**student\_id serial,**

**full\_name varchar,**

**grade int DEFAULT 1**

**);**

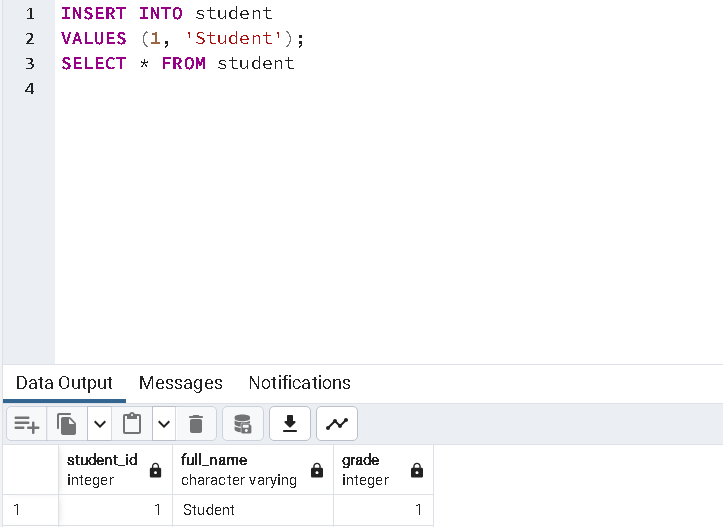


## **Задание 9**

Вставить запись в таблицу студентов и убедиться, что ограничение на вставку значения по умолчанию работает

**INSERT INTO student**

**VALUES (1, 'Student')**



## **Задание 10**

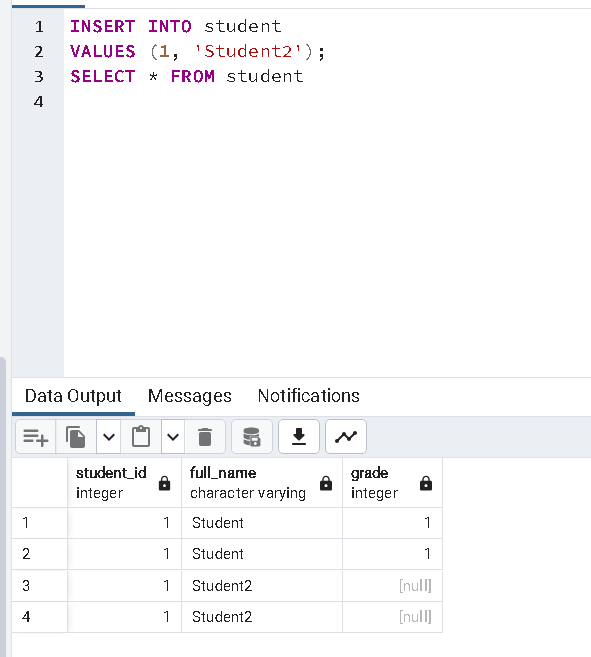
Удалить ограничение "по умолчанию" из таблицы студентов

**ALTER TABLE student**

**ALTER COLUMN grade DROP DEFAULT**

**INSERT INTO student**

**VALUES (1, 'Student2')**



## **Задание 11**

Подключиться к БД northwind и добавить ограничение на поле unit\_price таблицы products (цена должна быть больше 0)

**-- Добавление ограничения CHECK на поле unit\_price**

**ALTER TABLE products**

**ADD CONSTRAINT check\_unit\_price\_positive**

**CHECK (unit\_price > 0);**

## **Задание 12**

"Навесить" автоинкрементируемый счётчик на поле product\_id таблицы products (БД northwind). Счётчик должен начинаться с числа следующего за максимальным значением по этому столбцу.

**\*/**

**-- Находим максимальное значение product\_id в таблице products**

**SELECT MAX(product\_id) FROM products;**

**/\***

**Это запрос на нахождение максимального значения product\_id в таблице products. Он нужен, чтобы определить стартовое значение для автоинкрементного счётчика.**

**\*/**

**-- Создаем последовательность с именем products\_product\_id\_seq, начиная со следующего значения после максимального product\_id**

**CREATE SEQUENCE IF NOT EXISTS products\_product\_id\_seq**

**START WITH 79 OWNED BY products.product\_id;**

**/\***

**Это команда создаёт последовательность (sequence) с именем products\_product\_id\_seq, если такая последовательность ещё не существует.**

**Последовательность - это объект в базе данных, который генерирует уникальные числа в определённом порядке.**

**\*/**

**-- Обновляем поле product\_id в таблице products, чтобы оно использовало следующее значение из последовательности**

**ALTER TABLE products ALTER COLUMN product\_id SET DEFAULT nextval('products\_product\_id\_seq');**

**/\***

**В этой строке изменяется таблица products, чтобы использовать последовательность products\_product\_id\_seq в качестве значения по умолчанию для столбца product\_id.**

**Это гарантирует, что при вставке новых записей в таблицу, если явно не указано значение для product\_id, будет использоваться следующее значение из последовательности.**

**\*/**

**-- Вставка**

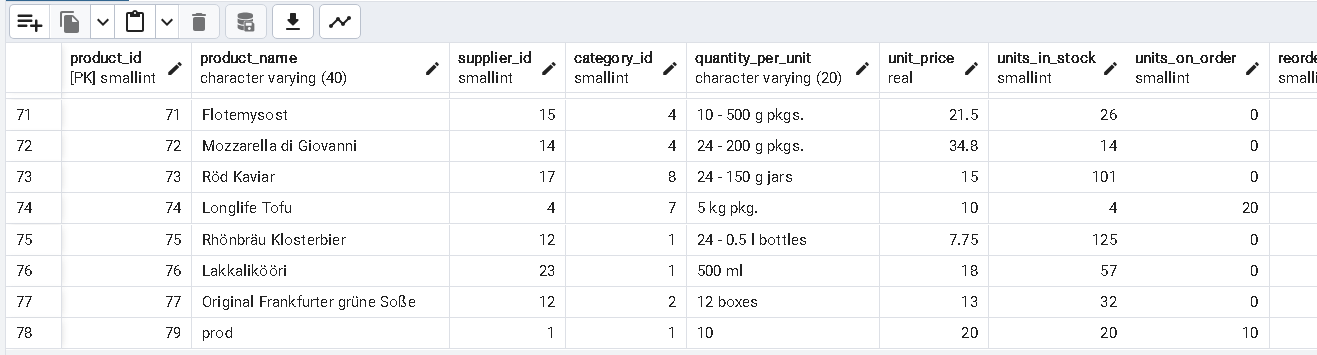
**INSERT INTO products(product\_name, supplier\_id, category\_id, quantity\_per\_unit, unit\_price, units\_in\_stock, units\_on\_order, reorder\_level, discontinued)**

**VALUES**

**('prod', 1, 1, 10, 20, 20, 10, 1, 0)**

**-- Находим максимальное значение product\_id в таблице products**

**SELECT MAX(product\_id) FROM products;**





## **Задание 13**

Произвести вставку в products (не вставляя идентификатор явно) и убедиться, что автоинкремент работает. Вставку сделать так, чтобы в результате команды вернулось значение, сгенерированное в качестве идентификатора.

**-- Вставка**

**INSERT INTO products(product\_name, supplier\_id, category\_id, quantity\_per\_unit, unit\_price, units\_in\_stock, units\_on\_order, reorder\_level, discontinued)**

**VALUES**

**('prod', 1, 1, 10, 20, 20, 10, 1, 0)**

# Заключение

В ходе выполнения курса по Postgre SQL мы ознакомились с основными принципами и возможностями этой базы данных. Мы научились создавать и управлять базами данных, проектировать таблицы, осуществлять операции CRUD (создание, чтение, обновление, удаление данных), работать с индексами и оптимизировать запросы.

Postgre SQL является мощной и гибкой системой управления базами данных, которая предлагает множество функциональных возможностей. Она обладает хорошей масштабируемостью и поддерживает широкий спектр типов данных. Благодаря своей открытой лицензии, Postgre SQL является свободной и бесплатной альтернативой коммерческим системам управления базами данных.

Одной из наиболее полезных функций Postgre SQL является поддержка транзакций, что обеспечивает целостность данных и защиту от сбоев в работе базы данных. Кроме того, Postgre SQL поддерживает многопользовательский режим, что позволяет нескольким пользователям одновременно работать с базой данных.

В процессе выполнения практических заданий мы приобрели навыки работы с различными SQL-командами, такими как SELECT, INSERT, UPDATE, DELETE и другими. Мы также научились создавать индексы, оптимизировать запросы и использовать различные методы для улучшения производительности базы данных.

В заключение можно сказать, что курс по Postgre SQL был очень полезным и позволил нам расширить наши знания в области управления базами данных. Мы теперь готовы применять полученные знания в нашей работе и помогать компаниям достичь эффективной работы с данными.

# Список используемой литературы.

PostgreSQL: Up and Running by Regina O. Obe, Leo S. Hsu

PostgreSQL Documentation: https://www.postgresql.org/docs/

PostgreSQL 13 Administration Cookbook by Simon Riggs, Gianni Ciolli, Ryan Lambert, Bruce Momjian

Learning PostgreSQL by Salah Mahmoud

Mastering PostgreSQL 12 by Hans-Jürgen Schönig