

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ**

**УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ**

**(национальный исследовательский университет)»**

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 311 «Прикладные программные средства и математические методы»

**КУРСОВАЯ РАБОТА**

**Дисциплина: «Информационные системы»**

**Тема: «Отчет по проделанной работе в PostgreSQL»**

Выполнил:

Студент гр. М3О-217Б-23

Белоус Кирилл Владимирович

Подпись:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Научный руководитель:

ассистент, каф. 311 Кос О.И.

Дата выдачи задания: 10.09.2024

Дата сдачи: -

Оценка ЛР:

Подпись руководителя:\_\_\_\_\_\_\_\_\_\_

Оглавление

[Лабораторная работа **1** 4](#_Toc152504920)

[Создание связи один к одному: 4](#_Toc152504921)

[Создание связи один ко многим: 5](#_Toc152504922)

[Создадим связь многие ко многим: 6](#_Toc152504923)

[Теперь заполним таблицу: 7](#_Toc152504924)

[Лабораторная работа **2** 7](#_Toc152504925)

[**Задание 2.1** 7](#_Toc152504926)

[**Задание 2.2.** 1](#_Toc152504927)0

[**Задание 2.3** 1](#_Toc152504928)2

[Лабораторная работа **3** 1](#_Toc152504929)6

[**Задание 1**](#_Toc152504930) 16

[**Задание 2**](#_Toc152504931) 17

[**Задание 3**](#_Toc152504932) 17

[**Задание 4**](#_Toc152504933) 17

[Лабораторная работа 4](#_Toc152504934) 17

[**Задание 1**](#_Toc152504935) 17

[**Задание 2**](#_Toc152504936) 18

[**Задание 3**](#_Toc152504937) 19

[**Задание 4** 2](#_Toc152504938)0

[Лабораторная](#_Toc152504939) работа 5 [2](#_Toc152504939)2

[**Часть 1** 2](#_Toc152504940)2

[**Задание 1** 2](#_Toc152504941)2

[**Задание 2** 2](#_Toc152504942)3

[**Задание 3** 2](#_Toc152504943)3

[**Задание 4** 2](#_Toc152504944)4

[**Задание 5** 2](#_Toc152504945)4

[**Задание 6** 2](#_Toc152504946)4

[**Задание 7** 2](#_Toc152504947)5

[**Задание 9** 2](#_Toc152504948)5

[**Часть 2** 2](#_Toc152504949)6

[**Задание 1** 2](#_Toc152504950)6

[**Задание 2** 2](#_Toc152504951)6

[**Задание 3** 2](#_Toc152504952)6

[**Задание 4** 2](#_Toc152504953)6

[**Задание 5**](#_Toc152504954) 27

[**Задание 6**](#_Toc152504955) 28

[**Задание 7**](#_Toc152504956) 28

[**Задание 8**](#_Toc152504957) 29

[**Задание 9**](#_Toc152504958) 29

[**Задание 10**](#_Toc152504959) 29

[**Задание 11**](#_Toc152504960) 30

[**Задание 12** 3](#_Toc152504961)0

[**Задание 13** 3](#_Toc152504962)1

[Лабораторная работа 7](#Лабораторная_7) 36

[**Задание 1**](#Лабораторная_7_1) 36

[**Задание 2**](#Лабораторная_7_2) 36

[**Задание 3**](#Лабораторная_7_3) 38

[Лабораторная работа 8](#Лабораторная_8) 40

[**Задание 1**](#Лабораторная_8_1) 40

[**Задание 2**](#Лабораторная_8_2) 41

[**Задание 3**](#Лабораторная_8_3) 41

[**Задание** 4](#Лабораторная_8_4) 42

[Заключение](#_Toc152504963) 43

[Список используемой литературы.](#_Toc152504964) 44

# Лабораторная работа 1

# Создание связи один к одному:

CREATE TABLE passport

(idd SERIAL PRIMARY KEY,

serial\_number varchar(12) NOT NULL);

CREATE TABLE person

(idd SERIAL PRIMARY KEY,

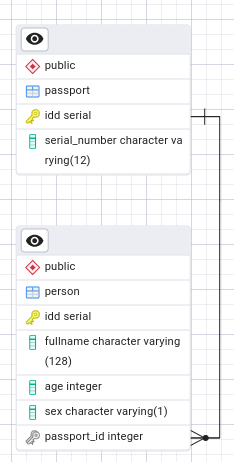
fullname varchar(128) NOT NULL,

age integer NOT NULL,

sex varchar(1) NOT NULL,

passport\_id integer NOT NULL UNIQUE,

constraint fk\_passport foreign key(passport\_id) references passport(idd))



Заполним таблицы данными:

INSERT INTO passport VALUES (DEFAULT, '880055-5353'), (DEFAULT, '556687-1337'),

(DEFAULT, '023455-5343');

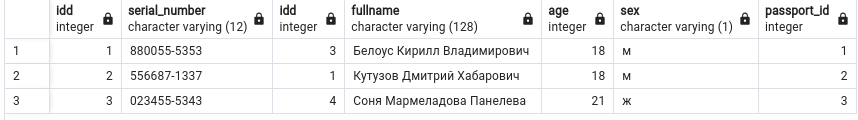
INSERT INTO person VALUES (DEFAULT, 'Кутузов Дмитрий Сергеевич', 18, 'м', 2),

(DEFAULT, 'Белоус Кирилл Владимирович', 18, 'м', 1),

(DEFAULT, 'Соня Мармеладова Панелева', 21, 'ж', 3);

Выводим результат на экран:

SELECT \* FROM passport JOIN person on passport.idd = person.passport\_id;



# Создание связи один ко многим:

CREATE TABLE publishers ( idd SERIAL PRIMARY KEY,

company\_name TEXT);

CREATE TABLE books(idd SERIAL PRIMARY KEY,

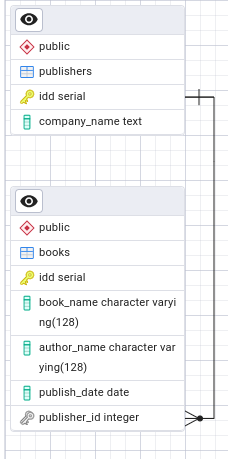
book\_name varchar(128) NOT NULL,

author\_name varchar(128) NOT NULL,

publish\_date DATE NOT NULL,

publisher\_id integer NOT NULL,

CONSTRAINT fk\_publisher FOREIGN KEY(publisher\_id) REFERENCES publishers(idd));

 INSERT INTO publishers VALUES (DEFAULT, 'chaihana'),

(DEFAULT, 'made in china'), (DEFAULT, 'pushkin dom');

INSERT INTO books VALUES (DEFAULT, 'Пожуй говна', 'Смирнов', '2019-05-02', 1),

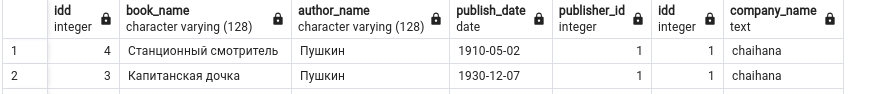
(DEFAULT, 'Введение в базы данных', 'Смирнов', '2020-02-03', 2),

(DEFAULT, 'Капитанская дочка', 'Пушкин', '1930-12-07', 1),

(DEFAULT, 'Станционный смотритель', 'Пушкин', '1910-05-02', 1),

(DEFAULT, 'Станционный смотритель', 'Пушкин', '1910-09-12', 2);

SELECT \* FROM books JOIN publishers ON books.publisher\_id = publishers.idd;



# Создадим связь многие ко многим:

CREATE TABLE author (

idd integer PRIMARY KEY,

full\_name TEXT

);

CREATE TABLE books

(

idd integer PRIMARY KEY,

book\_name varchar(128) NOT NULL,

publish\_date DATE NOT NULL

);

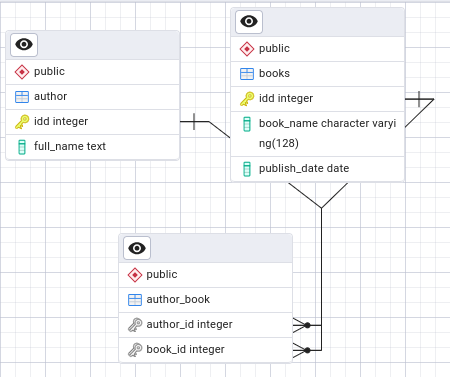
CREATE TABLE author\_book (

author\_id integer NOT NULL,

book\_id integer NOT NULL,

CONSTRAINT fk\_book FOREIGN KEY(book\_id) REFERENCES books(idd),

CONSTRAINT fk\_author FOREIGN KEY(author\_id) REFERENCES author(idd)

);

# Теперь заполним таблицу:

INSERT INTO author VALUES (1,'Пушкин'),(2,'Смирняга'),(3,'Яганов');

INSERT INTO books VALUES (1,'Станционный смотритель','1876-01-02'),

(2,'Как правильно шутить','2001-04-12'),

(3,'Почему использовать Винду это нормально?','2022-08-07'),

(4,'Колобок наносит ответный удар','2000-12-12');

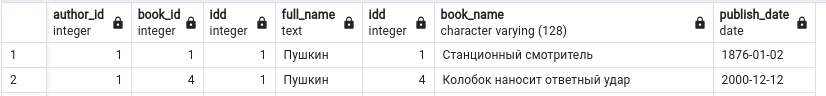
INSERT INTO author\_book VALUES (1,1),(1,4),(2,2),(3,3);

Выводим результат:

SELECT \* FROM author\_book

JOIN author ON author.idd = author\_book.author\_id

JOIN books ON books.idd = author\_book.book\_id ;

**

# Лабораторная работа 2

## **Задание 1**

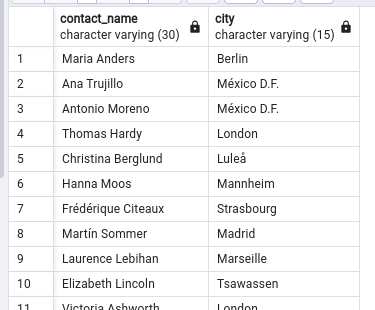
1. Выбрать все данные из таблицы customers

SELECT \* FROM customers;

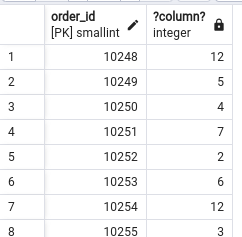


2. Выбрать все записи из таблицы customers, но только колонки "имя контакта" и "город"

SELECT contact\_name, city FROM customers;

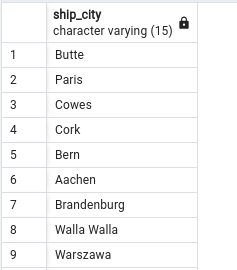
3. Выбрать все записи из таблицы orders, но взять две колонки: идентификатор заказа и колонку, значение в которой мы рассчитываем, как разницу между датой отгрузки и датой формирования заказа.

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



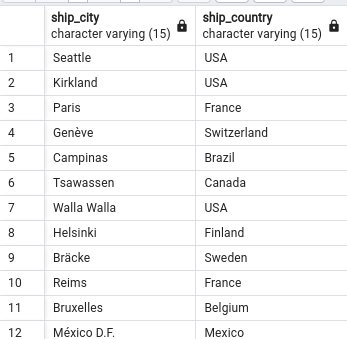
4. Выбрать все уникальные города, в которых "зарегистрированы" заказчики

SELECT DISTINCT ship\_city FROM orders;



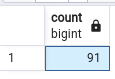
5. Выбрать все уникальные сочетания городов и стран, в которых "зарегистрированы" заказчики

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



6. Посчитать кол-во заказчиков

SELECT COUNT (\*) FROM customers;

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

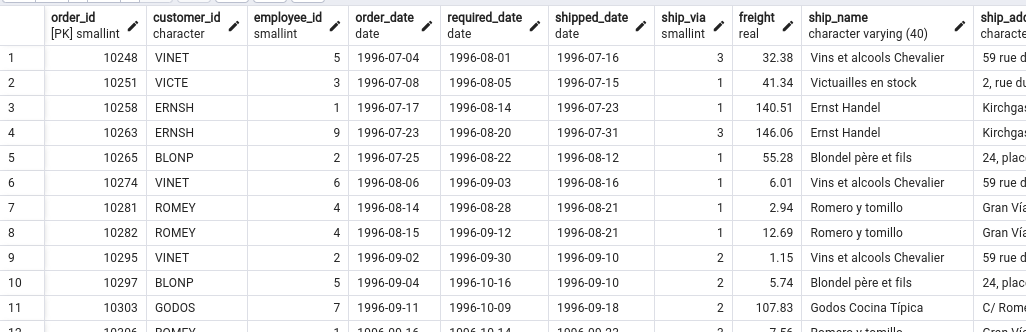
SELECT COUNT (DISTINCT ship\_country) FROM orders;

## **Задание 2.**

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

SELECT \* FROM orders

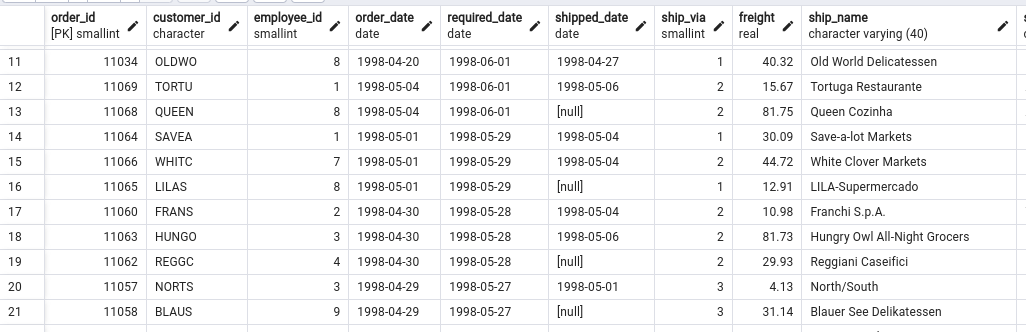
WHERE ship\_country IN ('France', 'Spain', 'Austria');



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.

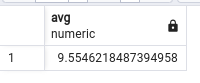
SELECT MAX (units\_in\_stock) FROM products WHERE units\_in\_stock > 30;



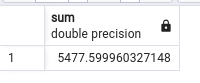
1. Найти среднее значение дней, уходящих на доставку с даты формирования заказа в USA

SELECT AVG (shipped\_date - order\_date)

FROM orders WHERE ship\_country = 'USA';



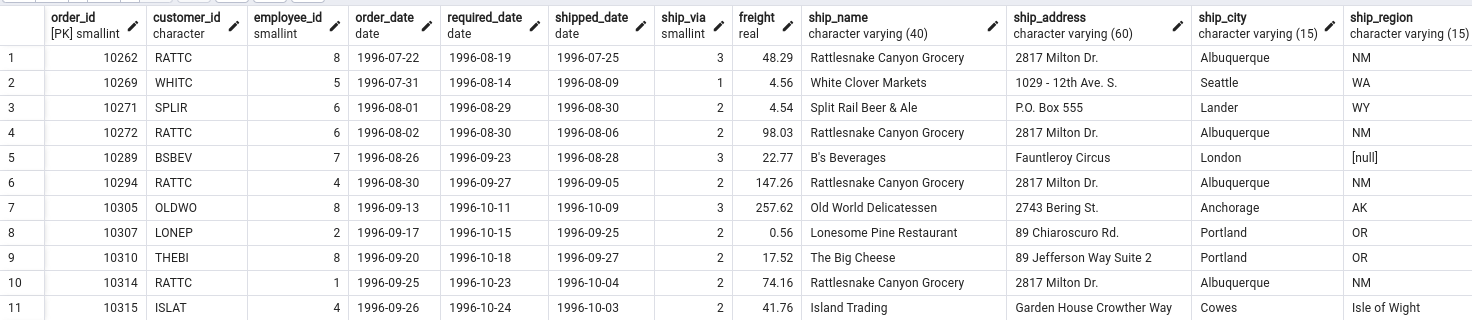
1. Найти сумму, на которую имеется товаров (кол-во \* цену) причём таких, которые планируется продавать и в будущем (см. на поле discontinued)

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

## **Задание3**

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%'

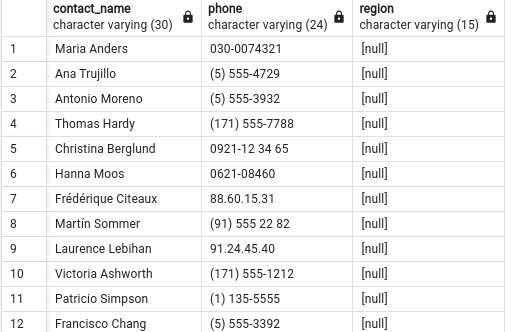
ORDER BY freight desc

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 not 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;

# 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,**

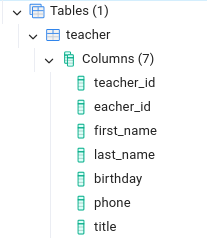
**eacher\_id serial,**

**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;**

## https://cdn.discordapp.com/attachments/1019238452555960331/1180813351668949032/image.png**Задание 3**

Удалить колонку middle\_name

**ALTER TABLE teacher**

**DROP COLUMN middle\_name;**

## 

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

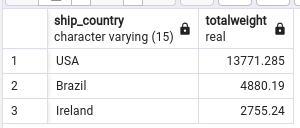
FROM orders

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;



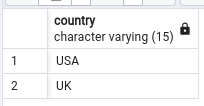
1. Выбрать такие страны, в которых "зарегистрированы" одновременно и заказчики, и поставщики, и работники.

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, first\_name, last\_name**

**FROM orders**

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

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

**INNER JOIN shippers ON shippers.shipper\_id = orders.ship\_via**

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

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

**AND shippers.company\_name = 'Speedy Express';**

## 

## **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 product\_name, units\_in\_stock, contact\_name, phone**

**FROM products**

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

**INNER JOIN categories ON products.category\_id = categories.category\_id**

**WHERE discontinued = '1'**

**AND (category\_name = 'Beverages' or category\_name = 'Seafood')**

**AND units\_in\_stock < 20;**

## **Задание 3**

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

**SELECT contact\_name, order\_id**

**FROM customers**

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

**WHERE 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 product\_name, units\_in\_stock**

**FROM products**

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

**SELECT MIN(avg\_quantity) As min\_avg**

**FROM (**

**SELECT AVG(quantity) AS avg\_quantity**

**FROM order\_details**

**GROUP BY product\_id**

**) AS averages**

**);**



## **`****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**

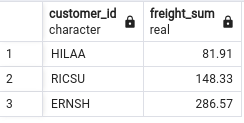
**WHERE shipped\_date BETWEEN '1996-7-15' AND '1996-8-1'**

**)**

**AND shipped\_date BETWEEN '1996-7-15' AND '1996-8-1'**

**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 order\_id, ship\_country, (**

**SELECT SUM(quantity \* unit\_price \* (1 - discount))**

**FROM order\_details**

**WHERE order\_details.order\_id = orders.order\_id) as order\_price**

**FROM orders**

**WHERE required\_date >= '1997-9-1'**

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

**ORDER BY order\_price DESC**

**LIMIT 3;**

## **Задание 4**

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

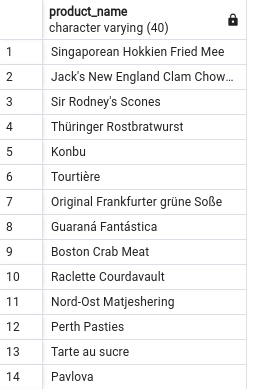
**SELECT (**

**SELECT product\_name**

**FROM products**

**WHERE products.product\_id = order\_details.product\_id) as p\_namFROM order\_details**

**WHERE 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,**

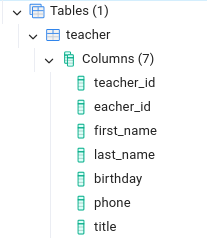
**last\_name varchar,**

**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;**

## https://cdn.discordapp.com/attachments/1019238452555960331/1180813351668949032/image.png**Задание 3**

Удалить колонку middle\_name

**ALTER TABLE teacher**

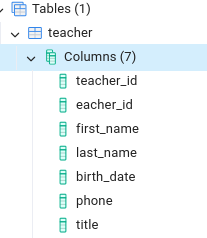
**DROP COLUMN middle\_name;**

## **Задание 4**

Переименовать колонку birthday в birth\_date

**ALTER TABLE teacher**

**RENAME birthday TO birth\_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

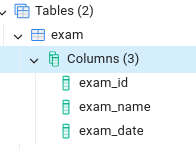
**CREATE TABLE exam (**

**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**

**('Дифференциальные уравнения', '2025-1-21'),**

**('Физика', '2025-1-25'),**

**('Алгоритмы и обработка данных', '2025-1-13');**

**SELECT \* FROM exam;**

## https://cdn.discordapp.com/attachments/1019238452555960331/1180813352683970621/image.png**Задание 9**

Удалить все данные из таблицы со сбросом идентификатор в исходное состояние

**TRUNCATE TABLE exam RESTART IDENTITY;**

**SELECT \* FROM exam;**

## 

## **Часть 2**

## **Задание 1**

Создать таблицу exam с полями:

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

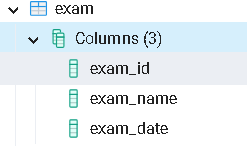
**CREATE TABLE exam (**

**exam\_id serial PRIMARY KEY,**

**exam\_name varchar,**

**exam\_date date**

**);**



## **Задание 2**

Удалить ограничение уникальности с поля идентификатора

**ALTER TABLE exam**

**DROP CONSTRAINT exam\_pkey;**

## **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**

**(**

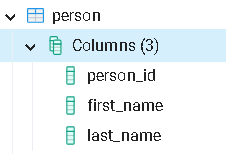
**person\_id integer NOT NULL,**

**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 integer,**

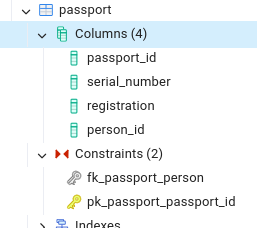
**serial\_number integer NOT NULL,**

**registration varchar(256),**

**person\_id integer,**

**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)

**CREATE TABLE book**

**(**

**book\_id int,**

**nm int**

**);**

**ALTER TABLE book**

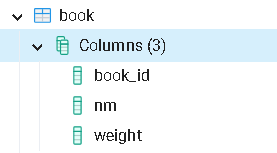
**ADD COLUMN weight integer CONSTRAINT chk\_book\_weight CHECK (weight > 0 AND weight < 100)**

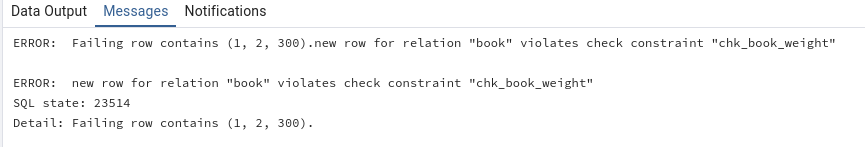
## **Задание 7**

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

**INSERT INTO book**

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

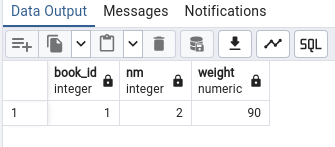




**INSERT INTO book**

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

**SELECT \* FROM book;**



## **Задание 8**

Создать таблицу student с полями:

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

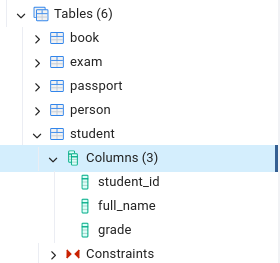
**CREATE TABLE student**

**(**

**student\_id serial,**

**full\_name varchar,**

**grade int DEFAULT 1**

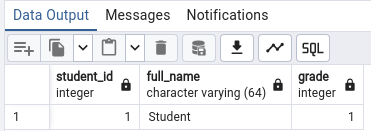
**);**

## **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, 'Кутузов Д.С');**

**SELECT \* FROM student;**

## **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.

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

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

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

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

## **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.

**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);**

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

# https://cdn.discordapp.com/attachments/1019238452555960331/1180817672057933865/image.png

# Лабораторная 7

## **Задание 1**

Создать представление, которое выводит следующие колонки:

order\_date, required\_date, shipped\_date, ship\_postal\_code, company\_name, contact\_name, phone, last\_name, first\_name, title из таблиц orders, customers и employees.

Сделать select к созданному представлению, выведя все записи, где order\_date больше 1го января 1997 года.

**CREATE VIEW orders\_customers\_employees AS**

**SELECT order\_date, required\_date, shipped\_date, ship\_postal\_code, company\_name, contact\_name, phone, last\_name, first\_name, title**

**FROM orders**

**JOIN customers USING(customer\_id)**

**JOIN employees USING(employee\_id);**

**SELECT \***

**FROM orders\_customers\_employees**

**WHERE order\_date > '1997-01-01';**

## 

## **Задание 2`**

Создать представление, которое выводит следующие колонки

order\_date, required\_date, shipped\_date, ship\_postal\_code, ship\_country, company\_name, contact\_name, phone, last\_name, first\_name, title из таблиц orders, customers, employees.

**CREATE VIEW orders\_customers\_employees AS**

**SELECT order\_date, required\_date, shipped\_date, ship\_postal\_code, ship\_country, company\_name, contact\_name, phone, last\_name, first\_name, title**

**FROM orders**

**JOIN customers USING(customer\_id)**

**JOIN employees USING(employee\_id);**

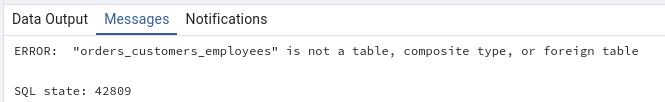
Попробовать добавить к представлению (после его создания) колонки postal\_code и reports\_to. Убедиться, что проихсодит ошибка.

**ALTER VIEW orders\_customers\_employees**

**ADD COLUMN postal\_code VARCHAR(20),**

**ADD COLUMN ship\_country VARCHAR(50),**

**ADD COLUMN reports\_to INT;**

Переименовать представление и создать новое уже с дополнительными колонками.

**ALTER VIEW orders\_customers\_employees**

**RENAME TO orders\_customers\_employees\_old;**

**CREATE VIEW orders\_customers\_employees AS**

**SELECT order\_date, required\_date, shipped\_date, ship\_postal\_code, ship\_country, company\_name, contact\_name, phone, last\_name, first\_name, title, customers.postal\_code, reports\_to**

**FROM orders**

**JOIN customers USING(customer\_id)**

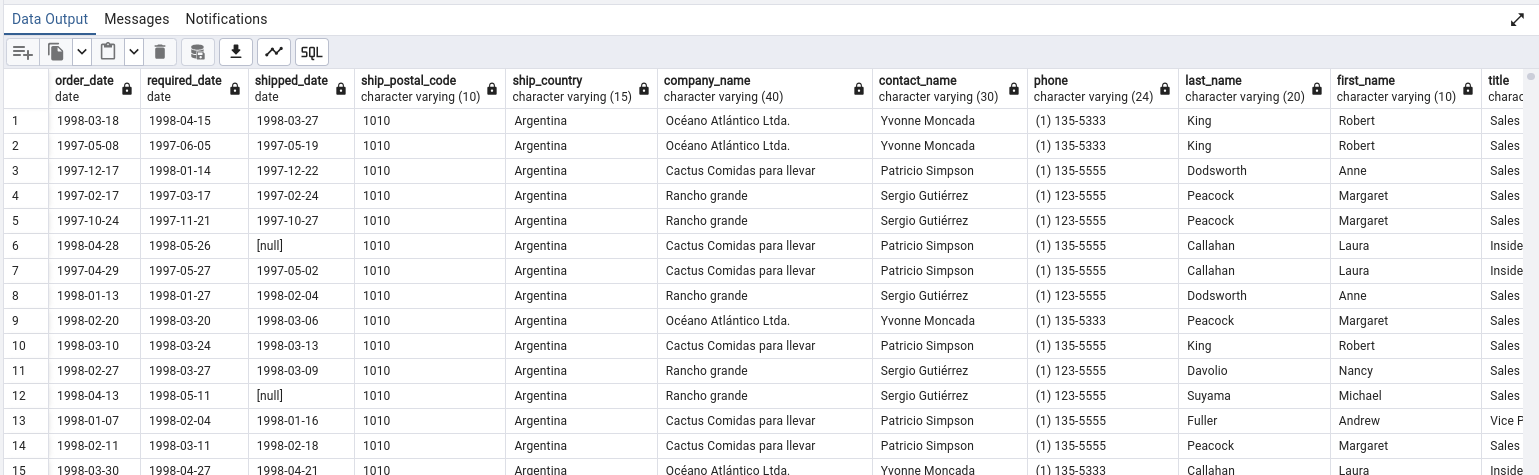
**JOIN employees USING(employee\_id);**

Сделать к нему запрос, выбрав все записи, отсортировав их по ship\_county.

**SELECT \***

**FROM orders\_customers\_employees**

**ORDER BY ship\_country;**



Удалить переименованное представление.

**DROP VIEW orders\_customers\_employees\_old;**

## **Задание 3**

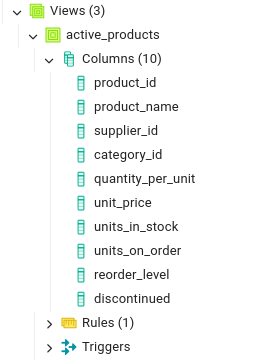
Создать представление "активных" (discontinued = 0) продуктов, содержащее все колонки. Представление должно быть защищено от вставки записей, в которых discontinued = 1

**CREATE VIEW active\_products AS**

**SELECT \***

**FROM products**

**WHERE discontinued = 0**

**WITH LOCAL CHECK OPTION;**

Попробовать сделать вставку записи с полем discontinued = 1 - убедиться, что не проходит.

**INSERT INTO active\_products**

**VALUES (1000, 'Пинок', 3, 4, 12, 20023, 120, 80, 5, 1);**

## **https://cdn.discordapp.com/attachments/1019238452555960331/1180817672057933865/image.png**

# Лабораторная 8

## **Задание 1**

Выполните следующий код (записи необходимы для тестирования корректности выполнения ДЗ):

insert into customers(customer\_id, contact\_name, city, country, company\_name)

values ('AAAAA', 'Alfred Mann', NULL, 'USA', 'fake\_company'),

('BBBBB', 'Alfred Mann', NULL, 'Austria','fake\_company');

После этого выполните задание:

Вывести имя контакта заказчика, его город и страну, отсортировав по возрастанию по имени контакта и городу,

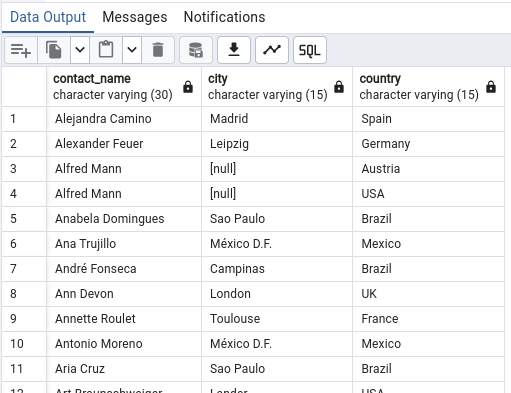
а если город равен NULL, то по имени контакта и стране. Проверить результат, используя заранее вставленные строки.

**SELECT contact\_name, city, country**

**FROM customers**

**ORDER BY**

**contact\_name ASC,**

**CO****ALESCE(city, country) ASC;**

## **Задание 2`**

Вывести наименование продукта, цену продукта и столбец со значениями

too expensive если цена >= 100

average если цена >=50 но < 100

low price если цена < 50

Попробовать добавить к представлению (после его создания) колонки postal\_code и reports\_to. Убедиться, что проихсодит ошибка.

**SELECT product\_name, unit\_price,**

**CASE WHEN unit\_price >= 100 THEN 'too expensive'**

**WHEN unit\_price >= 50 AND unit\_price < 100 THEN 'average'**

**WHEN unit\_price < 50 THEN 'low price'**

**ELSE 'undefined price'**

**END AS price\_rating**

**FROM products;**

**SELECT \***

**FROM orders\_customers\_employees**

**ORDER BY ship\_country;**

## **Задание 3**

Найти заказчиков, не сделавших ни одного заказа. Вывести имя заказчика и значение 'no orders' если order\_id = NULL.

**SELECT contact\_name,**

**CASE**

**WHEN order\_id IS NULL THEN 'no orders'**

**ELSE 'some orders'**

**END AS order\_existance**

**FROM customers**

**LEFT JOIN orders USING(customer\_id)**

**WHERE order\_id is NULL;**

## **Задание 4**

Вывести ФИО сотрудников и их должности. В случае если должность = Sales Representative вывести вместо неё Sales Stuff.

**SELECT first\_name, last\_name,**

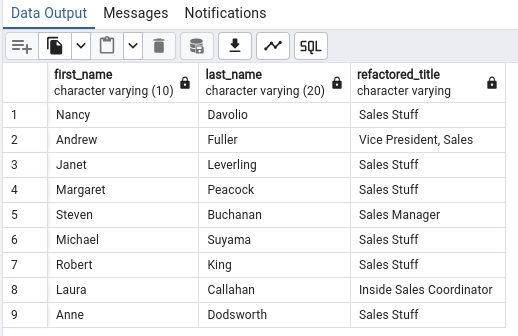
**CASE**

**WHEN title = 'Sales Representative' THEN 'Sales Stuff'**

**ELSE title**

**END AS refactored\_title**

**FROM employees**

****

# 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) и оптимизировать запросы. Также была изучена методология разработки базы данных, 1-3 НФ(Нормальные формы). Представления(view) и операци CRUD. Было разобранно ключевое отличие представления от таблиц, которое заключается в том, что view это виртуальная таблица, которая по сути не содержит данных . Мы также разобрали работу с CASE, IF.

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