

**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: «Создание и обработка запросов для базы данных для продуктовой компании “Яшкино”»**

Выполнила:

Студент гр. М3О-217Б-23

Белоус Кирилл Владимирович

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

Научный руководитель:

ассистент, каф. 311 Кос О.И. Дата выдачи задания: 15.11.2024 Дата сдачи: 13.12.2024

Оценка :

Подпись руководителя:

**Оглавление**

**Оглавление**

[Создание базы данных 3](#__RefHeading___Toc3123_2892134634)

[Создание таблиц и последовательностей 5](#__RefHeading___Toc3145_2892134634)

[Схема базы данных 9](#__RefHeading___Toc3147_2892134634)

[Заполнение таблиц данными. 11](#__RefHeading___Toc3153_2892134634)

[Запросы для базы данных 15](#__RefHeading___Toc3155_2892134634)

[Задание 2 15](#__RefHeading___Toc6903_1113697247)

[Задание 2.1 15](#__RefHeading___Toc3157_2892134634)

[Задание 2.2 16](#__RefHeading___Toc3165_2892134634)

[Задание 2.3 18](#__RefHeading___Toc3177_2892134634)

[Задание 3 19](#__RefHeading___Toc3179_2892134634)

[Задание 4 20](#__RefHeading___Toc3181_2892134634)

[Задание 7 22](#__RefHeading___Toc3189_2892134634)

[Задание №8 23](#__RefHeading___Toc3195_2892134634)

[Выводы 26](#__RefHeading___Toc3197_2892134634)

[Список литературы 27](#__RefHeading___Toc6905_1113697247)

# Создание базы данных

### Связи между таблицами:

### 1. warehouses ↔ warehouse\_product

* Тип связи: Один-ко-многим (один склад может содержать множество продуктов, но каждый продукт связан с конкретным складом в таблице warehouse\_product).
* Ключевые поля:
  + warehouses.warehouse\_id ↔ warehouse\_product.warehouse\_id.

### 2. products ↔ categories

* Тип связи: Один-ко-многим (одна категория может включать множество продуктов, но каждый продукт относится только к одной категории).
* Ключевые поля:
  + categories.category\_id ↔ products.category\_id.

### 3. products ↔ warehouse\_product

* Тип связи: Один-ко-многим (один продукт может находиться на нескольких складах, но на одном складе продукт хранится в определённом количестве).
* Ключевые поля:
  + products.product\_id ↔ warehouse\_product.product\_id.

### 4. products ↔ order\_details

* Тип связи: Один-ко-многим (один продукт может быть включён в несколько заказов, но каждая запись в order\_details связана с конкретным продуктом).
* Ключевые поля:
  + products.product\_id ↔ order\_details.product\_id.

### 5. customer\_company ↔ customers

* Тип связи: Один-ко-многим (одна компания может иметь нескольких заказчиков, но каждый заказчик принадлежит только одной компании).
* Ключевые поля:
  + customer\_company.company\_id ↔ customers.company\_id.

### 6. customers ↔ orders

* Тип связи: Один-ко-многим (один заказчик может сделать множество заказов, но каждый заказ связан с конкретным заказчиком).
* Ключевые поля:
  + customers.customer\_id ↔ orders.customer\_id.

### 7. warehouses ↔ orders

* Тип связи: Один-ко-многим (один склад может быть связан с несколькими заказами, но каждый заказ относится к конкретному складу).
* Ключевые поля:
  + warehouses.warehouse\_id ↔ orders.warehouse\_id.

### 8. shippers ↔ orders

* Тип связи: Один-ко-многим (один поставщик может обрабатывать множество заказов, но каждый заказ связан с конкретным поставщиком).
* Ключевые поля:
  + shippers.shipper\_id ↔ orders.shipper\_id.

### 9. supplier\_companies

* Особенность: Эта таблица изолирована, в текущей схеме не указаны связи с другими таблицами. Обычно такая таблица используется для указания поставщиков продуктов.

# **Создание таблиц и последовательностей**

1)Создадим все необходимые таблицы:

Создание таблицы warehouses(складов)

create table warehouses(

warehouse\_id serial primary key,

warehouse\_name varchar,

location varchar

);

Создание таблицы categories(категориев)

create table categories(

category\_id serial primary key,

category\_name varchar,

description varchar

);

Создание таблицы shippers(поставщики)

create table shippers(

shipper\_id serial primary key,

company\_name varchar,

phone varchar,

freight integer

);

Создание таблицы products(продукты)

create table products(

product\_id serial primary key,

product\_name varchar,

category\_id integer,

quantity\_per\_unit integer,

unit\_price integer,

CONSTRAINT fk\_category FOREIGN KEY (category\_id) REFERENCES categories(category\_id)

);

Создание таблицы warehouse\_product(продукты на складе и их кол-во)

create table warehouse\_product(

warehouse\_id integer,

product\_id integer,

stock\_quantity varchar,

constraint fk\_product foreign key (product\_id) references products(product\_id),

constraint fk\_warehouse foreign key (warehouse\_id) references warehouses(warehouse\_id)

);

Создание таблицы supplier\_companies(компании и владельцы компании поставщиков)

create table supplier\_companies(

company\_id serial primary key,

company\_name varchar,

addres varchar,

city varchar,

country varchar,

phone varchar

);

Создание таблицы order\_details(детали заказов)

create table order\_details(

order\_id serial primary key,

product\_id integer,

quantity integer,

constraint fk\_product foreign key (product\_id) references products(product\_id)

);

Создание таблицы customer\_company(компании заказчика)

create table customer\_company(

company\_id serial primary key,

company\_name varchar,

address varchar,

city varchar,

country varchar,

phone varchar

);

Создание таблицы customers(заказчиков)

create table customers(

customer\_id serial primary key,

contact\_name varchar,

company\_id integer,

constraint fk\_company foreign key (company\_id) references customer\_company(company\_id)

);

Создание таблицы orders(заказов)

create table orders(

order\_id integer,

customer\_id integer,

warehouse\_id integer,

warehouse\_product\_id integer,

shipper\_id integer,

order\_date date,

order\_required date,

constraint fk\_customer foreign key (customer\_id) references customers(customer\_id),

constraint fk\_warehouses foreign key (warehouse\_id) references warehouses(warehouse\_id),

constraint fk\_shipper foreign key (shipper\_id) references shippers(shipper\_id)

);

# Схема базы данных

1. Таблицы находятся в первой нормальной форме (1NF):
   * Все значения в таблицах атомарны (неделимы). Например, в каждой колонке хранится только одно значение.
   * Все строки уникальны благодаря наличию первичных ключей.
2. Таблицы находятся во второй нормальной форме (2NF):
   * Каждая таблица имеет первичный ключ.
   * Все неключевые атрибуты зависят от всего первичного ключа, а не от его части. Например, в таблице warehouse\_product связь между warehouse\_id и product\_id (первичный составной ключ) гарантирует, что stock\_quantity зависит от обоих ключей.
3. Таблицы находятся в третьей нормальной форме (3NF):
   * Нет транзитивной зависимости между неключевыми атрибутами.  
     Например:
     + В таблице products колонка unit\_price зависит только от product\_id, а не от внешнего ключа category\_id.
     + В таблице customers колонка contact\_name зависит только от customer\_id, а не от внешнего ключа company\_id.
   * В каждой таблице все неключевые атрибуты зависят только от первичного ключа и не зависят друг от друга.

### Разделение на отдельные таблицы:

* Таблицы связаны внешними ключами (FOREIGN KEY), что предотвращает избыточность данных. Например:
  + Название категории (category\_name) и её описание (description) находятся в таблице categories, а не дублируются в products.
  + Информация о складах (warehouse\_name и location) находится в таблице warehouses, а не дублируется в таблицах о запасах или заказах.
* Каждая таблица отвечает за свою сущность (например, товары, склады, категории, поставщики, заказчики).

### Преимущества:

* Отсутствие избыточности: Дублирование сведений минимизировано благодаря нормализации.
* Логическая согласованность: Изменение данных в одном месте не приводит к необходимости изменять их в других таблицах.
* Простота обновления: Обновление, добавление и удаление данных выполняется без риска нарушения целостности.

Таким образом, структура базы данных соответствует требованиям 3NF, обеспечивает эффективное хранение данных и упрощает работу с ними.

# Заполнение таблиц данными.

2)Теперь заполним таблицы данными:

Таблица warehouses

INSERT INTO warehouses (warehouse\_name, location)

VALUES

('Главный склад', 'Москва'),

('Склад Запад', 'Калининград'),

('Склад Восток', 'Владивосток'),

('Склад Север', 'Санкт-Петербург'),

('Склад Юг', 'Краснодар');

Таблица categories

INSERT INTO categories (category\_name, description)

VALUES

('Сладости', 'Шоколад, конфеты, печенье'),

('Выпечка', 'Пирожные, торты, кексы'),

('Снэки', 'Чипсы, сухарики, орехи'),

('Напитки', 'Соки, вода, чай'),

('Молочные продукты', 'Молоко, сыр, йогурты');

Таблица shippers

INSERT INTO shippers (company\_name, phone, freight)

VALUES

('Доставка+','+7 495 123-45-67', 1000),

('ЭкспрессКарго','+7 812 987-65-43', 1500),

('РегионТранс','+7 343 567-89-01', 1200),

('СибирьЛогистик','+7 385 123-45-67', 2000),

('ЮгКарго','+7 861 987-65-43', 1700);

Таблица products

INSERT INTO products (product\_name, category\_id, quantity\_per\_unit, unit\_price)

VALUES

('Шоколад Яшкино', 1, 24, 50),

('Печенье Яшкино', 1, 12, 40),

('Торт Яшкино', 2, 6, 300),

('Кекс Яшкино', 2, 8, 150),

('Сухарики Яшкино', 3, 20, 30),

('Орехи Яшкино', 3, 15, 100),

('Чай Яшкино', 4, 10, 200),

('Сок Яшкино', 4, 12, 120),

('Йогурт Яшкино', 5, 6, 60),

('Молоко Яшкино', 5, 10, 50);

Таблица warehouse\_product

INSERT INTO warehouse\_product (warehouse\_id, product\_id, stock\_quantity)

VALUES

(1, 1, '500'),

(1, 2, '300'),

(2, 3, '200'),

(2, 4, '150'),

(3, 5, '400'),

(3, 6, '250'),

(4, 7, '100'),

(4, 8, '120'),

(5, 9, '300'),

(5, 10, '400');

Таблица supplier\_companies

INSERT INTO supplier\_companies (company\_name, addres, city, country, phone)

VALUES

('Кондитерский Дом Яшкино', 'ул. Центральная, 15', 'Киров', 'Россия', '+7 8332 45-67-89'),

('Продукты Яшкино', 'ул. Ленина, 10', 'Пермь', 'Россия', '+7 342 12-34-56'),

('Сладости Яшкино', 'ул. Гагарина, 5', 'Ижевск', 'Россия', '+7 3412 98-76-54'),

('ВкусВилл Яшкино', 'ул. Советская, 20', 'Самара', 'Россия', '+7 846 56-78-90'),

('Северный Склад', 'ул. Комсомольская, 3', 'Архангельск', 'Россия', '+7 8182 45-67-89');

Таблица order\_details

INSERT INTO order\_details (product\_id, quantity)

VALUES

(1, 50),

(2, 30),

(3, 10),

(4, 15),

(5, 60),

(6, 40),

(7, 25),

(8, 20),

(9, 35),

(10, 50);

Таблица customer\_company

INSERT INTO customer\_company (company\_name, address, city, country, phone)

VALUES

('Магнит', 'ул. Ленина, 1', 'Москва', 'Россия', '+7 495 123-45-67'),

('Пятерочка', 'ул. Тверская, 10', 'Санкт-Петербург', 'Россия', '+7 812 987-65-43'),

('Ашан', 'ул. Центральная, 15', 'Казань', 'Россия', '+7 843 123-45-67'),

('Перекресток', 'ул. Победы, 5', 'Екатеринбург', 'Россия', '+7 343 567-89-01'),

('Лента', 'ул. Гоголя, 20', 'Новосибирск', 'Россия', '+7 383 987-65-43');

Таблица customers

INSERT INTO customers (contact\_name, company\_id)

VALUES

('Иванов Иван', 1),

('Петров Петр', 2),

('Сидоров Сидор', 3),

('Кузнецов Алексей', 4),

('Попова Анна', 5);

Таблица orders

INSERT INTO orders (order\_id, customer\_id, warehouse\_id, shipper\_id, order\_date, order\_required)

VALUES

(1, 1, 1, 1, '2024-12-01', '2024-12-05'),

(2, 2, 2, 2, '2024-12-02', '2024-12-06'),

(3, 3, 3, 3, '2024-12-03', '2024-12-07'),

(4, 4, 4, 4, '2024-12-04', '2024-12-08'),

(5, 5, 5, 5, '2024-12-05', '2024-12-09');

# Запросы для базы данных

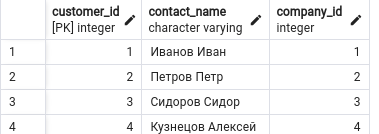
Теперь повторим все запросы, которые мы реализовали в лабораторных работах (2,3,4,7,8).

## Задание 2

## Задание 2.1

**Выбрать всех клиентов**

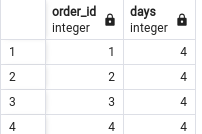
select \* from customers;



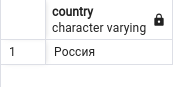
### Выбрать идентификатор заказа и разницу между датой формирования заказа и текущей датой

select order\_id, order\_required - order\_date as days

from orders;



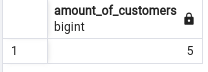
### Выбрать все уникальные регионы, в которых зарегистрированы заказчики



Посчитать количество клиентов

select count(\*) as amount\_of\_customers

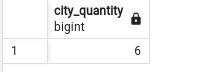
from customers;



### Посчитать количество уникальных городов, в которых зарегистрированы клиенты

select count(distinct city) as city\_quantity

from customer\_company;



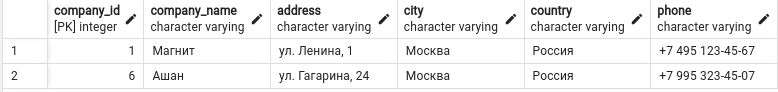
## Задание 2.2

### Выбрать все заказы клиентов из города "Москва"

select \*

from customer\_company

where city = 'Москва';



### Выбрать все заказы, отсортировав по убыванию даты заказа

select \*

from orders

order by order\_date desc;



### Выбрать минимальную цену продукта среди продуктов с ценой выше 100

select min(unit\_price) as min\_price

from products

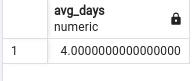
where unit\_price > 100;



### Среднее количество дней между текущей датой и датой заказа.

select avg(order\_required - order\_date) as avg\_days

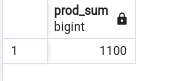
from orders;



### Сумма стоимости всех товаров

select sum(unit\_price) as prod\_sum

from products;

****

## Задание 2.3

Выбрать все компании, где название компании начинается начинается с "Ма"

select \*

from customer\_company

where company\_name like 'Ма%';



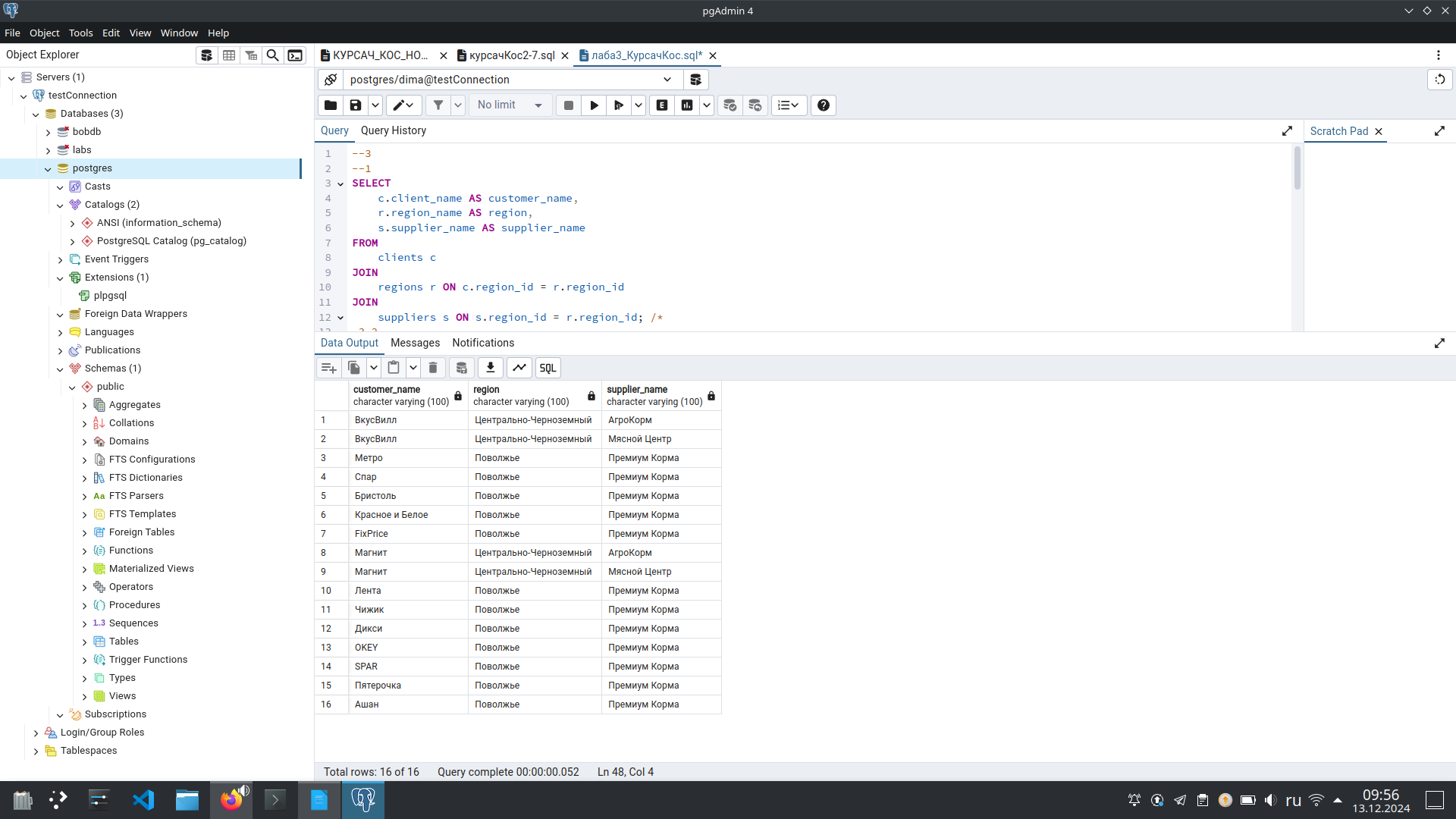
## Задание 3

**1. Найти заказчиков которые которые СДЕЛАЛИ заказ.**

select contact\_name, order\_id

from customers

JOIN orders using(customer\_id);



**2. Найти заказчиков, НЕ СДЕЛАВШИХ ни одного заказа. Вывести имя заказчика и order\_id.**

select contact\_name, order\_id

from customers

left JOIN orders using(customer\_id)

where order\_id is NULL;

****

## Задание 4

### **1. Получить список продуктов средняя цена больше средней цены всех продуктов в той же категории. При этом нужно вывести: название продукта, категорию, склад и количество на складе.**

SELECT

p.product\_name AS product,

c.category\_name AS category,

w.warehouse\_name AS warehouse,

wp.stock\_quantity AS stock

FROM warehouse\_product wp

JOIN products p ON wp.product\_id = p.product\_id

JOIN warehouses w ON wp.warehouse\_id = w.warehouse\_id

JOIN categories c ON p.category\_id = c.category\_id

WHERE p.unit\_price > (SELECT AVG(p2.unit\_price)

FROM products p2

WHERE p2.category\_id = p.category\_id)

ORDER BY p.product\_name;



**3. Получить список складов, на которых хранится хотя бы один продукт из категории с названием "Electronics", и общее количество таких продуктов на каждом складе.**

SELECT w.warehouse\_name AS warehouse, COUNT(wp.product\_id) AS total\_products FROM warehouses w

JOIN warehouse\_product wp ON w.warehouse\_id = wp.warehouse\_id

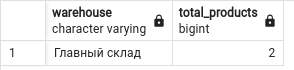
JOIN products p ON wp.product\_id = p.product\_id

JOIN categories c ON p.category\_id = c.category\_id

WHERE c.category\_name = 'Electronics'

GROUP BY w.warehouse\_id, w.warehouse\_name

ORDER BY total\_products DESC;



## Задание 7

## **1.Создать представление, которое выводит данные о заказах, включая дату заказа и информацию о клиентах (название клиента и регион)**

## 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';

**2. Создать представление с информацией о заказах и клиентах, добавить новые колонки и убедиться, что обновление представления невозможно**

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);

ALTER VIEW orders\_customers\_employees

ADD COLUMN postal\_code VARCHAR(20),

ADD COLUMN ship\_country VARCHAR(50),

ADD COLUMN reports\_to INT;

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);

**3. Создать представление "дорогих" продуктов, где unit\_price >= 100.**

CREATE VIEW active\_products AS

SELECT \*

FROM products

WHERE discontinued = 0

WITH LOCAL CHECK OPTION;

## Задание №8

**1. Вывести имя заказчика (client\_name) и регион, отсортировав по имени заказчика и региону.**

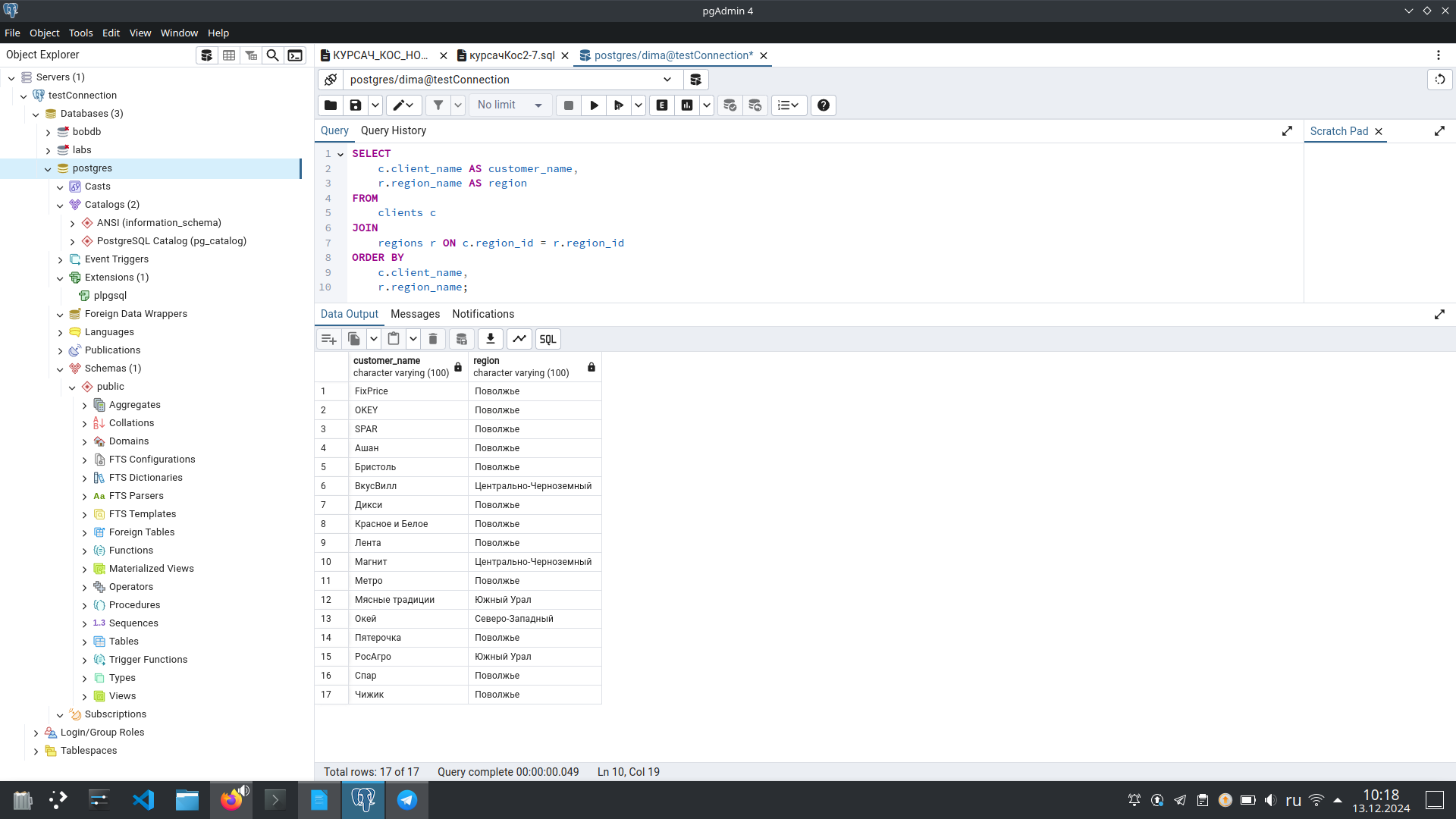
SELECT contact\_name, city, country

FROM customers

ORDER BY

contact\_name ASC,

COALESCE(city, country) ASC;



**2.** Вывести наименование продукта, цену продукта и столбец со значениями:

* too expensive, если цена >= 500;
* average, если цена от 50 до 99;
* low price, если цена < 50.

SELECT product\_name, unit\_price,

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

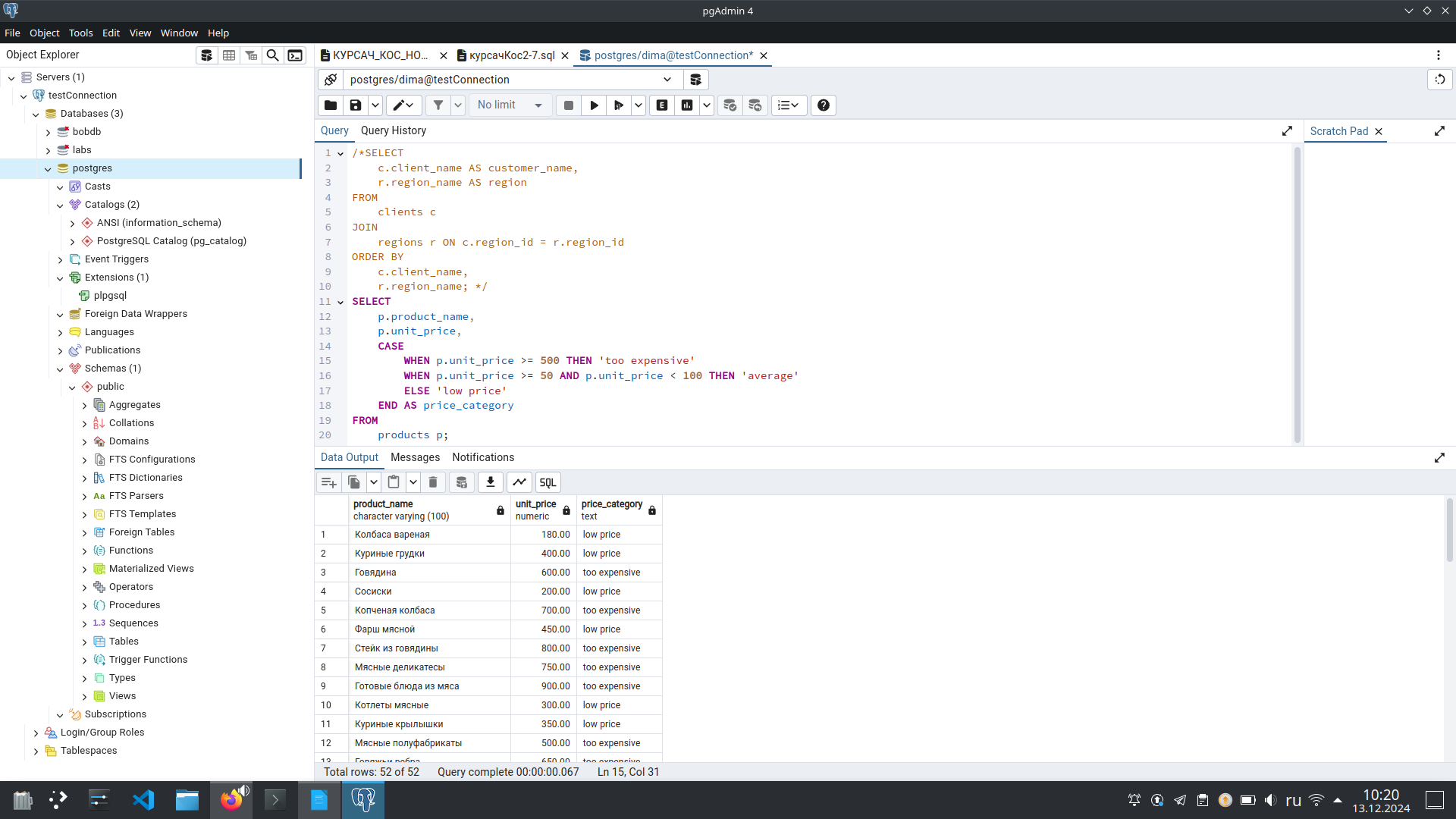
WHEN unit\_price >= 50 AND unit\_price < 100 THEN 'average'

WHEN unit\_price < 50 THEN 'low price'

ELSE 'undefined price'

END AS price\_rating

FROM products;



**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;



**4. Найти поставщиков (suppliers) и добавить столбец, где для поставщиков из региона "Поволжье" будет указано "Local Supplier", а для всех остальных "Non-local Supplier".**

SELECT first\_name, last\_name,

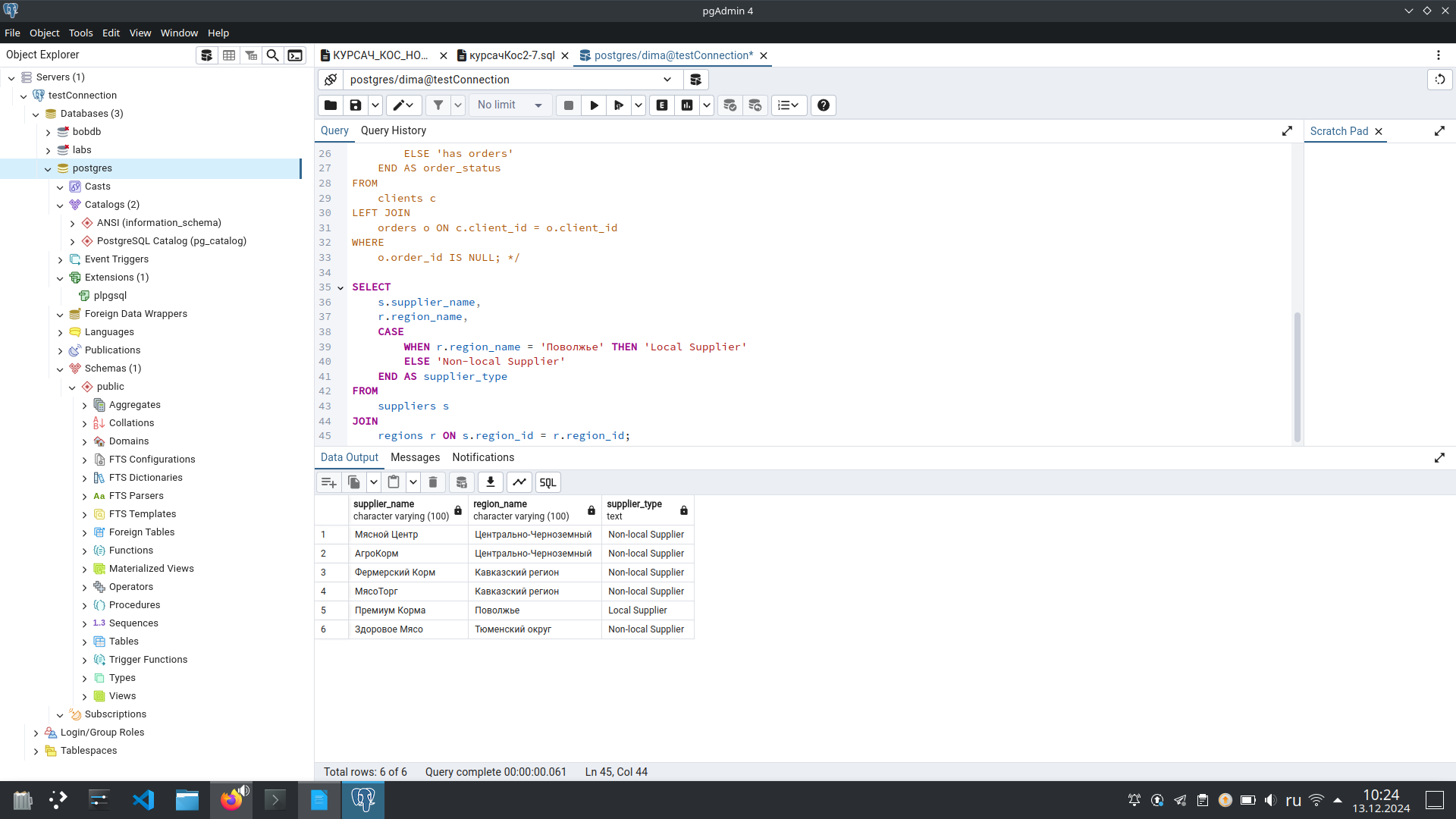
CASE

WHEN title = 'Local' THEN 'Non local'

ELSE title

END AS refactored\_title

FROM suppliers



# Выводы

В процессе выполнения курсовой работы по курсу PostgreSQL мы изучили основные принципы проектирования баз данных и освоили базовые запросы в этой среде. В результате нашей работы была разработана база данных для кондитрской фабрики “Яшкино”, в котором были созданы таблицы и установлены связи между ними. Затем мы применили запросы, которые изучали в ходе курса, к этой базе данных, выбирали из БД все поля, некоторые поля, поля подходящие под условие. Так же мы объединяли таблицы, смотрели их пересечение объединение, выводили всю левую таблицу, всю правую таблицу, делали подзапросы. Создавали представления(виртуальные таблицы), создавали последовательности, заменяли в уже существующих таблицах последовательности. Так же применили поиск строк где поле начинается с какой-то буквы. В итоге мы приобрели навыки работы в гибкой и мощной системе управления базами данных, научились создавать базы данных, заполнять их информацией и выполнять различные запросы различной сложности к ним.

# Список литературы

1. PostgreSQL: Up and Running by Regina O. Obe, Leo S. Hsu
2. PostgreSQL Documentation: https://www.postgresql.org/docs/
3. PostgreSQL 13 Administration Cookbook by Simon Riggs, Gianni Ciolli, Ryan Lambert, Bruce Momjian