МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: "Информационная безопасность / Information security"**

**Дисциплина:**

**«*Информационная безопасность баз данных*»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙРАБОТЕ № 2**

***«Манипулирование данными в БД на языке SQL»***

**Выполнил студент(ы):**

группа/поток 1.3

Бардышев Артём Антонович/\_\_\_\_\_\_\_\_\_\_

*Подпись*

**Проверил:**

Карманова Наталья Андреевна/\_\_\_\_\_\_\_\_\_\_

*Подпись*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата*

Санкт-Петербург

2025г.

1. **Цель работы:** Получение навыков манипулирования данными в БД при помощи операторов SQL.
2. **Теоретическая информация:**

Особенности хранения данных в БД накладывает ограничения на манипулирование этими данными. Для эффективного решения задач, связанных с получением нужной информации из БД существует специальных язык структурированных запросов в БД – SQL. Все запросы оформляются в виде операторов SQL, обладающих собственным синтаксисом. В зависимости от реализации СУБД реализация SQL запросов может несколько отличаться, но базовые примитивы остаются неизменными. Наряду со стандартными операциями работы с данными – вставка, удаление, извлечение, обновление, SQL обладает расширенными возможностями по работе с данными, позволяющими получать данные в соответствии с разнообразными условиями. Владение навыками работы с БД при помощи SQL позволяет эффективно решать задачи по систематизации, поиску, хранению информации в СУБД.

В качестве СУБД, используемой в лабораторной работе, предполагается PostgreSQL. Основные операторы представлены ниже.

**Оператор SELECT**

В общем виде оператор SELECT конструируется в виде:

[WITH *запросы\_with*] SELECT *список\_выборки* FROM *табличное\_выражение* [WHERE *условие\_ограничения*][*определение\_сортировки*]

Где WITH подразумевает возможность использования результата другого оператора SQL, например, SELECT, с которым уже производится работа, как с таблицей в текущем запросе. Список выборки - перечень атрибутов, которые должны быть отображены в полученной таблице, указываются атрибуты, присутствующие в табличном выражении, порядок указания значения не имеет. В качестве табличного выражения выступает таблица, которая может вычисляться на этапе выполнения оператора. Обычная таблица представляет собой физическую таблицу в БД, но при необходимости можно преобразовывать и комбинировать на основе физических таблиц и получать виртуальную таблицу. Условие ограничения - это любое выражение, выдающие результат в виде boolean, которое применяется к каждой строке полученной таблицы из табличного выражения. Если результат вычисление true, то такая строка остается в противном случае отбрасывается. Определение сортировки позволяет упорядочить строки по значению одного из атрибутов.

Пример:

*SELECT sensor\_name FROM sensor\_translation WHERE maсh\_id=65540 AND lang\_id=1 AND sensor\_no=0 AND group\_name='E1'*

**Оператор INSERT**

[WITH *запросы\_with*] INSERT INTO *таблица (список\_атрибутов) VALUES* (список\_значений)

Добавление новых записей в таблицу производится оператором INSERT. Необходимо в явном виде указать таблицу, в которую добавляются строки, а также список атрибутов добавляемой записи и ее значения.

Пример:

*INSERT INTO gps (gps\_id,utc\_datetime,maсh\_id,long\_value,lat\_value,speed) VALUES (132,'2016-01-04 14:00',60031,4545.9090,7878.9090,0);*

**Оператор UPDATE**

[WITH *запросы\_with*] UPDATE *таблица SET атрибут=новое значение* [WHERE *условие\_обновления*]

Для обновления записи в таблице используется оператор UPDATE. В качестве параметров необходимо указывать атрибуты, которые необходимо обновить и новые значения этих атрибутов после ключевого слова SET. Для обновления только определенных атрибутов существует возможность применять условие обновления после ключевого слова WHERE с тем же синтаксисом что в операторе SELECT.

Пример:

*UPDATE machines SET mach\_type=7 WHERE mach\_id=65537;*

**Оператор DELETE**

[WITH *запросы\_with*] DELETE FROM *таблица* [WHERE *условие\_удаления*]

Оператор DELETE позволяет удалять записи из таблицы. При необходимости можно использовать условие удаления с ключевым словом WHERE.

Пример:

*DELETE FROM report WHERE type=7;*

Все перечисленные операторы имеют возможности по использованию дополнительных параметров, с которыми необходимо ознакомиться в документации.

1. **Задание**
2. Создайте по крайней мере 3 связанные таблицы. Должны быть определены первичные и внешние ключи для таблиц, т.е. по крайней мере для одной пары таблиц должна быть определена связь 1:M
3. Наполнить таблицы базы данных при помощи операторов INSERT. Каждая таблица должна иметь не менее 5 разных записей.
4. Обновить записи в одной таблице на основании записи из другой (между таблицами должна быть связь).
5. Удалить несколько записей из одной таблицы на основании информации из другой таблицы.
6. Вывести часть столбцов из таблицы.
7. Вывести несколько записей из таблицы, используя условие ограничения.
8. Сделать декартово произведение двух таблиц.
9. Вывести записи из таблицы на основании условия ограничения, содержащегося в другой таблице.
10. Применить функции агрегирования к выводимым записям (sum, avg, min, max)
11. Вывести записи из таблицы, используя сортировку от большего к меньшему.
12. Вывести записи из таблицы, используя сортировку от меньшего к большему с ограничением количества выводимых строк.
13. Произвести агрегирование выводимых записей по одному из полей ( group by).
14. Выполнить запрос, когда табличное выражение представляет собой другой запрос.

ХОД РАБОТЫ:

Мы создадим **5 таблиц**:

1. **customers** — Клиенты магазина.
2. **orders** — Заказы, сделанные клиентами.
3. **products** — Товары (iPhone, MacBook, AirPods и т. д.).
4. **order\_items** — Какие товары включены в заказ.
5. **categories** — Категории товаров (iPhone, MacBook, iPad и др.).

1) Создание таблиц

-- 1. Таблица клиентов

CREATE TABLE customers (

customer\_id SERIAL PRIMARY KEY,

name VARCHAR(100) NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

phone VARCHAR(15) UNIQUE NOT NULL

);

-- 2. Таблица заказов (связана с customers)

CREATE TABLE orders (

order\_id SERIAL PRIMARY KEY,

customer\_id INT REFERENCES customers(customer\_id) ON DELETE CASCADE,

order\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

total\_price DECIMAL(10,2) DEFAULT 0 CHECK (total\_price >= 0)

);

-- 3. Таблица категорий товаров (iPhone, MacBook и т. д.)

CREATE TABLE categories (

category\_id SERIAL PRIMARY KEY,

category\_name VARCHAR(50) UNIQUE NOT NULL

);

-- 4. Таблица товаров (связана с categories)

CREATE TABLE products (

product\_id SERIAL PRIMARY KEY,

category\_id INT REFERENCES categories(category\_id) ON DELETE CASCADE,

product\_name VARCHAR(100) NOT NULL,

price DECIMAL(10,2) NOT NULL CHECK (price > 0),

stock\_quantity INT NOT NULL CHECK (stock\_quantity >= 0)

);

-- 5. Таблица товаров в заказе (связана с orders и products)

CREATE TABLE order\_items (

item\_id SERIAL PRIMARY KEY,

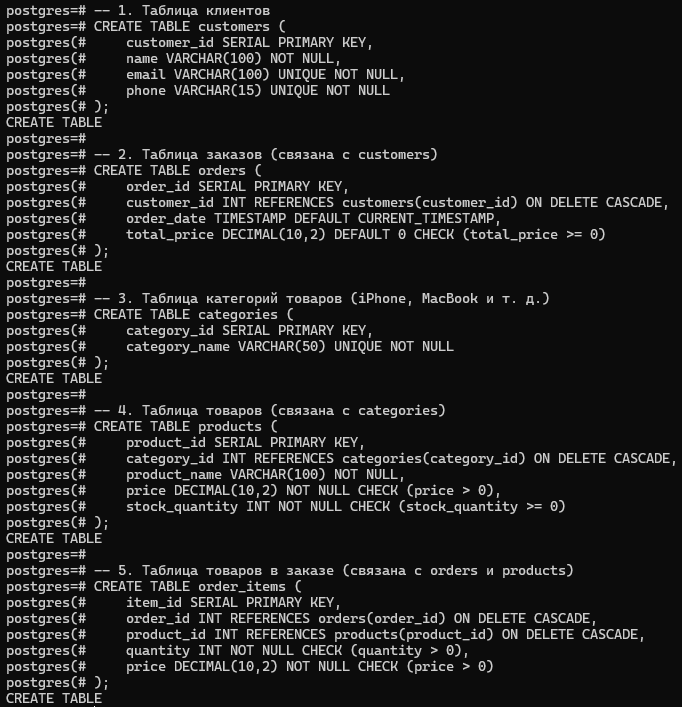
order\_id INT REFERENCES orders(order\_id) ON DELETE CASCADE,

product\_id INT REFERENCES products(product\_id) ON DELETE CASCADE,

quantity INT NOT NULL CHECK (quantity > 0),

price DECIMAL(10,2) NOT NULL CHECK (price > 0)

);



2) Наполнение таблиц данными

INSERT INTO customers (name, email, phone) VALUES

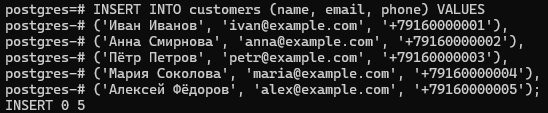
('Иван Иванов', 'ivan@example.com', '+79160000001'),

('Анна Смирнова', 'anna@example.com', '+79160000002'),

('Пётр Петров', 'petr@example.com', '+79160000003'),

('Мария Соколова', 'maria@example.com', '+79160000004'),

('Алексей Фёдоров', 'alex@example.com', '+79160000005');



Добавляем категории товаров

INSERT INTO categories (category\_name) VALUES

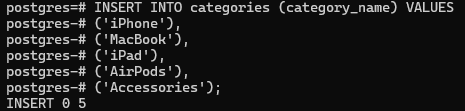
('iPhone'),

('MacBook'),

('iPad'),

('AirPods'),

('Accessories');



Добавляем товары

INSERT INTO products (category\_id, product\_name, price, stock\_quantity) VALUES

-- iPhone

(1, 'iPhone 15 Pro Max', 139990.00, 10),

(1, 'iPhone 15 Pro', 129990.00, 15),

(1, 'iPhone 15', 99990.00, 20),

(1, 'iPhone 14', 89990.00, 30),

-- MacBook

(2, 'MacBook Pro 16 M3', 299990.00, 5),

(2, 'MacBook Air 13 M2', 149990.00, 10),

-- iPad

(3, 'iPad Pro 12.9 M2', 129990.00, 8),

(3, 'iPad Air 10.9', 79990.00, 12),

-- AirPods

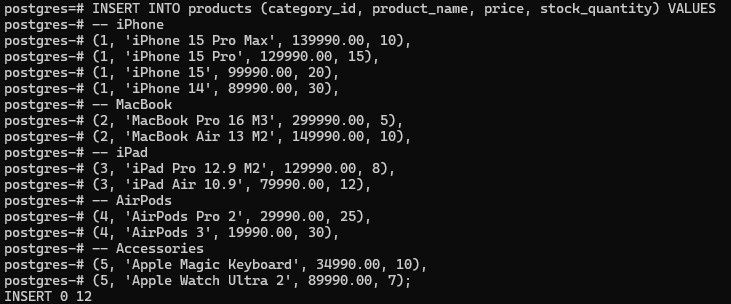
(4, 'AirPods Pro 2', 29990.00, 25),

(4, 'AirPods 3', 19990.00, 30),

-- Accessories

(5, 'Apple Magic Keyboard', 34990.00, 10),

(5, 'Apple Watch Ultra 2', 89990.00, 7);



Добавляем заказы

INSERT INTO orders (customer\_id, order\_date) VALUES

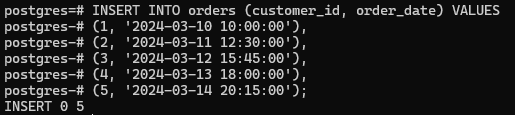
(1, '2024-03-10 10:00:00'),

(2, '2024-03-11 12:30:00'),

(3, '2024-03-12 15:45:00'),

(4, '2024-03-13 18:00:00'),

(5, '2024-03-14 20:15:00');



Добавляем товары в заказы

INSERT INTO order\_items (order\_id, product\_id, quantity, price) VALUES

(1, 1, 1, 139990.00), -- iPhone 15 Pro Max

(1, 10, 1, 19990.00), -- AirPods 3

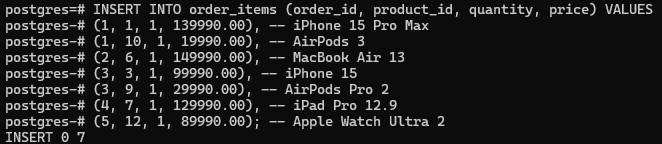
(2, 6, 1, 149990.00), -- MacBook Air 13

(3, 3, 1, 99990.00), -- iPhone 15

(3, 9, 1, 29990.00), -- AirPods Pro 2

(4, 7, 1, 129990.00), -- iPad Pro 12.9

(5, 12, 1, 89990.00); -- Apple Watch Ultra 2



3) Вывести часть столбцов

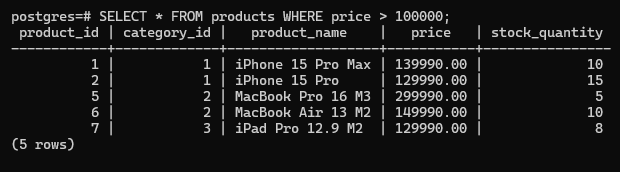
SELECT product\_name, price FROM products;



4) Вывести несколько записей с условием

SELECT \* FROM products WHERE price > 100000;

SELECT \* FROM products WHERE price > 100000;



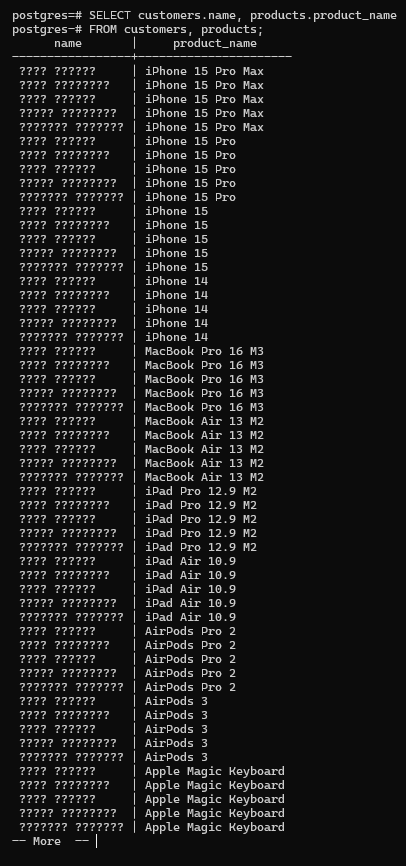
5) Вывод части столбцов

SELECT name, email FROM customers;

6) **Декартово произведение**

SELECT customers.name, products.product\_name

FROM customers, products;

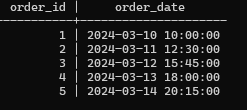


7) Вывести записи из одной таблицы, используя данные из другой

SELECT customers.name, orders.order\_id, orders.order\_date

FROM customers

JOIN orders ON customers.customer\_id = orders.customer\_id;

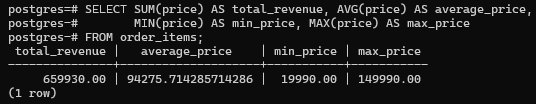


8) Агрегатные функции (SUM, AVG, MIN, MAX)

SELECT SUM(price) AS total\_revenue, AVG(price) AS average\_price,

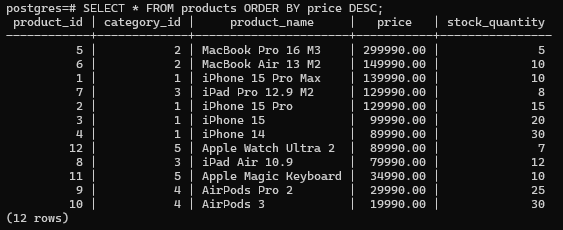
MIN(price) AS min\_price, MAX(price) AS max\_price

FROM order\_items;



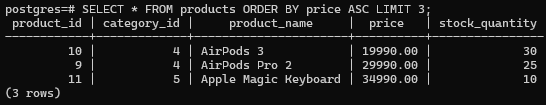
9) Сортировка от большего к меньшему

SELECT \* FROM products ORDER BY price DESC;



10) Сортировка от меньшего к большему с ограничением

SELECT \* FROM products ORDER BY price ASC LIMIT 3;



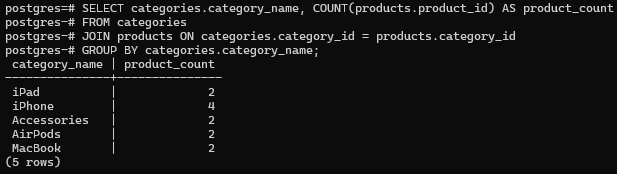
11) Группировка по категориям

SELECT categories.category\_name, COUNT(products.product\_id) AS product\_count

FROM categories

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

GROUP BY categories.category\_name;



12) Использование WITH

WITH order\_summary AS (

SELECT order\_id, SUM(price \* quantity) AS total\_order\_price

FROM order\_items

GROUP BY order\_id

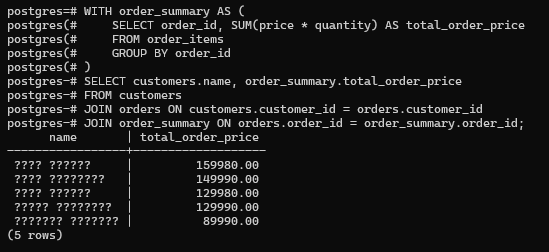
)

SELECT customers.name, order\_summary.total\_order\_price

FROM customers

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

JOIN order\_summary ON orders.order\_id = order\_summary.order\_id;



Вывод:

Я проделал полный цикл работы с базой данных **Apple Store** в PostgreSQL, начиная с проектирования структуры, наполнения данными, выполнения сложных SQL-запросов и заканчивая удалением всех данных.