

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

ОТЧЕТ

по Лабораторной работе № 4

**«Запросы на выборку и модификацию данных. Представления. Работа с
индексами»**

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Ермаков Максим Олегович

Факультет прикладной информатики

Группа К3240

Направление подготовки 09.03.03 Прикладная информатика

Образовательная программа Мобильные и сетевые технологии 2023

Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2025

СОДЕРЖАНИЕ

Оглавление

| | |
|---|-----------|
| ВВЕДЕНИЕ..... | 3 |
| 1 Схема базы данных..... | 4 |
| 2 Выполнение | 5 |
| 2.1 Запросы к базе данных..... | 5 |
| 2.2 Представления..... | 9 |
| 2.3 Запросы на модификацию данных | 12 |
| 2.4 Создание индексов..... | 17 |
| ВЫВОД | 20 |

ВВЕДЕНИЕ

Цель работы - овладеть практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.

Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию лабораторной работы №2, часть 2 и 3).
2. Составить 3 запроса на модификацию данных (INSERT, UPDATE, DELETE) **с использованием подзапросов**.
3. Изучить графическое представление запросов и просмотреть историю запросов.
4. Создать простой и составной индексы для двух произвольных запросов и сравнить время выполнения запросов без индексов и с индексами. Для получения плана запроса использовать команду EXPLAIN.

1 Схема базы данных

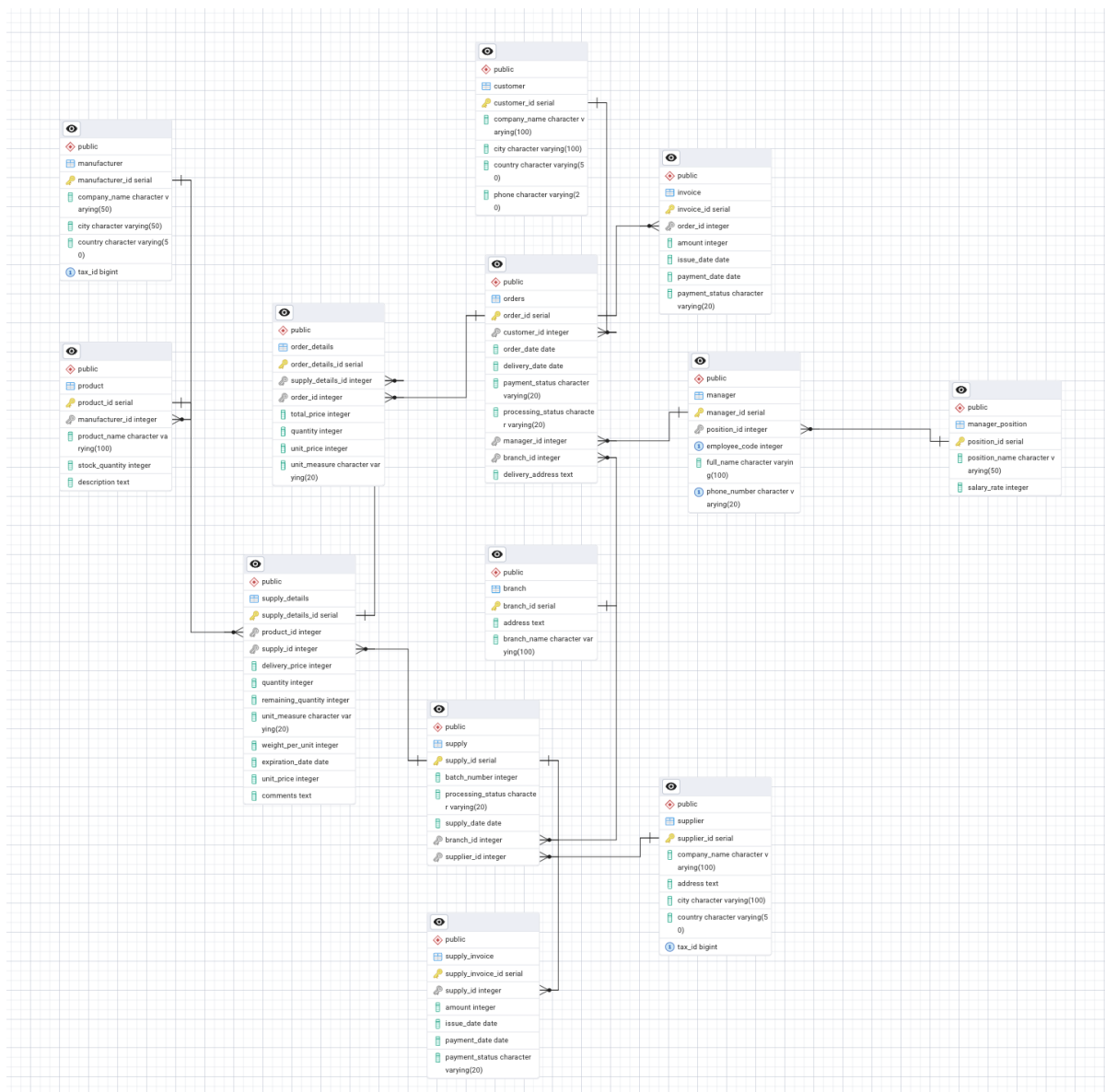


Рисунок 1 – Схема логической модели базы данных, сгенерированная в Generate ERD

2 Выполнение

2.1 Запросы к базе данных

1. Формулировка запроса: Вывести список поставщиков, которые поставляют все товары.

```
SELECT s.supplier_id, s.company_name
FROM supplier s
WHERE NOT EXISTS (
    SELECT 1
    FROM product p
    WHERE NOT EXISTS (
        SELECT 1
        FROM supply_details sd
        JOIN supply sp ON sd.supply_id = sp.supply_id
        WHERE sd.product_id = p.product_id AND sp.supplier_id = s.supplier_id
    )
);
```

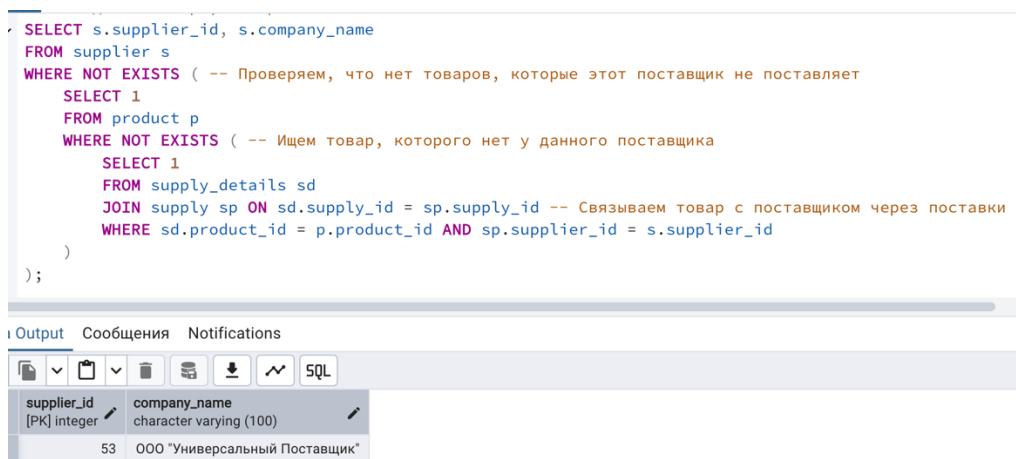


Рис.2 – Скриншот выполнения запроса 1 с результатом

2. Формулировка запроса: Найти поставщиков, которые поставляют каждый из товаров по самой низкой цене.

```
SELECT DISTINCT s.supplier_id, s.company_name, sd.product_id, sd.unit_price
FROM supply_details sd
JOIN supply sp ON sd.supply_id = sp.supply_id
JOIN supplier s ON sp.supplier_id = s.supplier_id
JOIN (
    SELECT product_id, MIN(unit_price) AS min_price
    FROM supply_details
```

GROUP BY product_id
) AS min_prices
 ON sd.product_id = min_prices.product_id AND sd.unit_price = min_prices.min_price;

потом найдем тех поставщиков, которые по этой цене действительно поставляют:

```

SELECT DISTINCT s.supplier_id, s.company_name, sd.product_id, sd.unit_price
FROM supply_details sd
JOIN supply sp ON sd.supply_id = sp.supply_id -- Привязываем детали поставки к самой поставке
JOIN supplier s ON sp.supplier_id = s.supplier_id -- Привязываем поставщика к поставке
JOIN (
  SELECT product_id, MIN(unit_price) AS min_price
  FROM supply_details
  GROUP BY product_id -- Вычисляем минимальную цену для каждого товара
) AS min_prices
ON sd.product_id = min_prices.product_id AND sd.unit_price = min_prices.min_price; -- Оставляем
  
```

Output Сообщения Notifications

| supplier_id integer | company_name character varying (100) | product_id integer | unit_price integer |
|------------------------|---|-----------------------|-----------------------|
| 1 | ИП "БиоПродукт-869" | 13 | 23 |
| 2 | ИП "Урожай-649" | 22 | 17 |
| 2 | ИП "Урожай-649" | 27 | 20 |
| 3 | ИП "Фермер-562" | 44 | 17 |
| 4 | ЗАО "СнабжениеРегион-248" | 41 | 21 |
| 6 | ООО "Поставкин-667" | 26 | 19 |
| 7 | ЗАО "ОвощМаркет-726" | 16 | 23 |
| 7 | ЗАО "ОвощМаркет-726" | 36 | 18 |
| 9 | АО "БиоПродукт-137" | 19 | 18 |
| 10 | АО "Фермер-795" | 42 | 24 |
| 10 | АО "Фермер-795" | 47 | 23 |
| 11 | ООО "ПродуктТрейд-214" | 31 | 22 |
| 12 | АО "Поставкин-608" | 29 | 13 |

Рисунок 3 – Скриншот выполнения запроса 2 с результатом

3. Формулировка запроса: Вывести названия товаров, цены на которые у всех поставщиков одинаковы.

```

SELECT p.product_name
FROM product p
JOIN supply_details sd ON p.product_id = sd.product_id
GROUP BY p.product_id, p.product_name
HAVING MIN(unit_price) = MAX(unit_price);
  
```

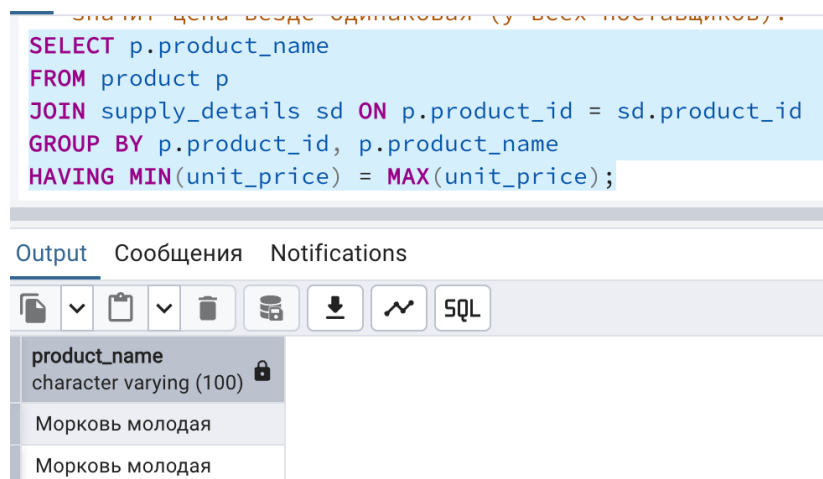


Рисунок 4 – Скриншот выполнения запроса 3 с результатом

4. Формулировка запроса: Чему равен общий суточный доход оптового склада за 2024-03-18?

```

SELECT SUM(od.total_price) AS daily_income
FROM order_details od
JOIN orders o ON od.order_id = o.order_id
WHERE o.order_date = DATE '2024-03-18';

```

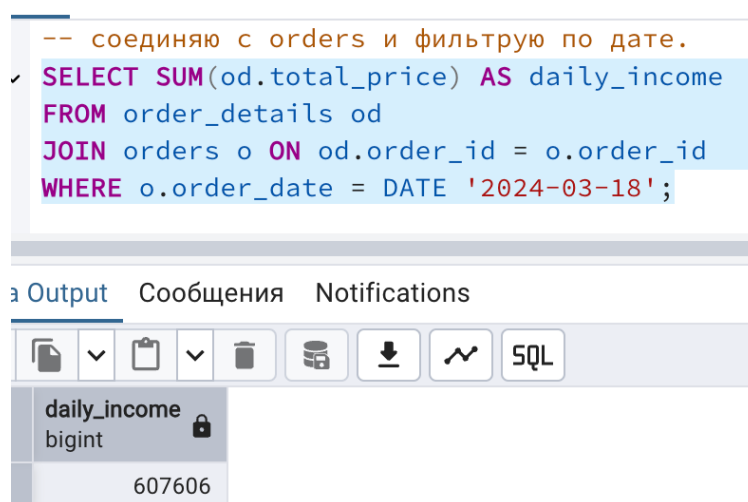


Рисунок 5 – Скриншот выполнения запроса 4 с результатом

5. Формулировка запроса: Вычислить общую стоимость каждого вида товара, находящегося на базе.

SELECT

```

p.product_name,
SUM(sd.remaining_quantity * sd.unit_price) AS total_value
FROM supply_details sd
JOIN product p ON sd.product_id = p.product_id
GROUP BY p.product_name;

```

-- Логика: считаю сумму остатка товара * цену, группирую по pro

```

SELECT
  p.product_name,
  SUM(sd.remaining_quantity * sd.unit_price) AS total_value
FROM supply_details sd
JOIN product p ON sd.product_id = p.product_id
GROUP BY p.product_name;

```

Output Сообщения Notifications

SQL

| product_name character varying (100) | total_value bigint |
|---|-----------------------|
| Лук репчатый | 551487 |
| Брусника | 580019 |
| Клюква | 626992 |
| Дыня | 532896 |
| Лимоны | 567868 |
| Баклажаны | 522946 |
| Персики | 756359 |
| Капуста краснокочанная | 799961 |
| Фасоль стручковая | 516033 |
| Редис | 605010 |
| Лук-порей | 461972 |
| Помидоры | 519164 |
| Сливы | 492796 |
| Картофель | 715703 |
| Хурма | 488576 |

Рисунок 6 – Скриншот выполнения запроса 5 с результатом

6.Формулировка запроса: В какой день было вывезено минимальное количество товара?

```

SELECT order_date, SUM(od.quantity) AS total_quantity
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY order_date
ORDER BY total_quantity ASC
LIMIT 1;

```

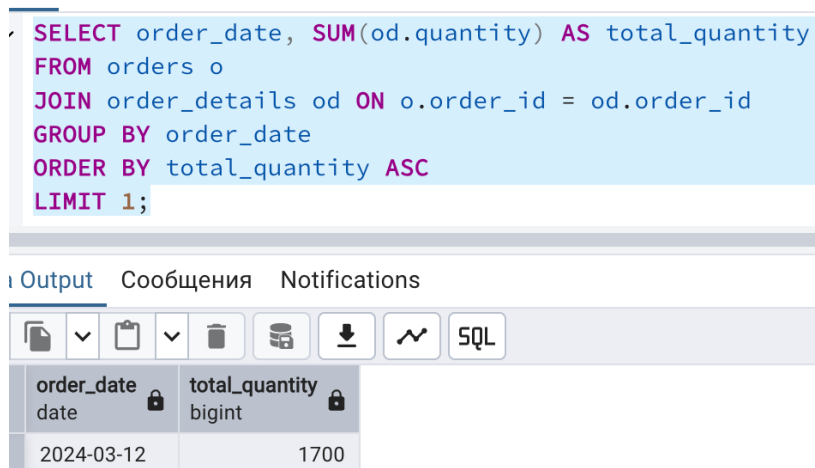



Рисунок 7 – Скриншот выполнения запроса 6 с результатом

7. Формулировка запроса: Сколько различных видов товара имеется на базе?

```
SELECT COUNT(DISTINCT product_id) AS products_in_stock
FROM supply_details
WHERE remaining_quantity > 0;
```

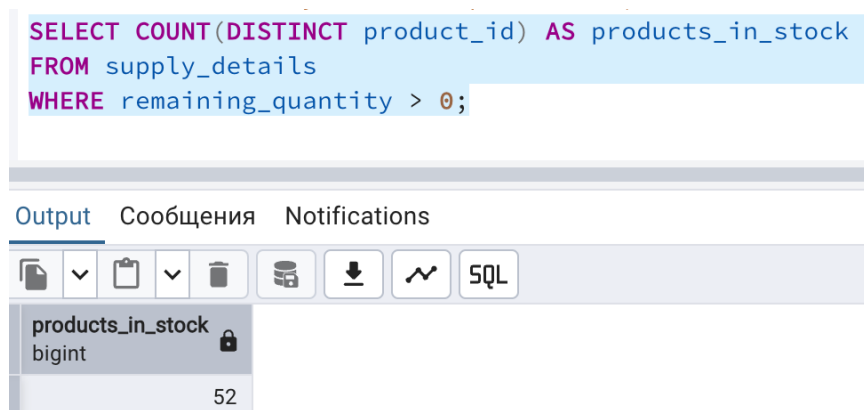


Рисунок 8 – Скриншот выполнения запроса 7 с результатом

2.2 Представления

1. Формулировка запроса: количество заказов фирм-покупателей за прошедший год;

```
CREATE VIEW view_customer_order_count_year AS
SELECT
  c.customer_id,
  c.company_name,
  COUNT(o.order_id) AS order_count
FROM customer c
LEFT JOIN orders o ON c.customer_id = o.customer_id
WHERE o.order_date BETWEEN DATE '2024-03-01' AND DATE '2024-03-31'
GROUP BY c.customer_id, c.company_name;
```

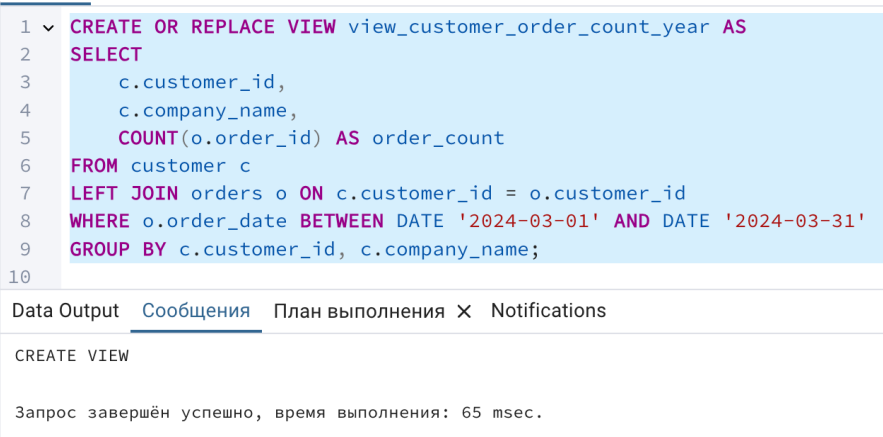


Рисунок 9 – Скриншот создания представления 1

The screenshot shows the SQL Developer interface with a query executed. The top pane shows the query: 'SELECT * FROM view_customer_order_count_year;'. The bottom pane shows the results in a table format. The table has three columns: 'customer_id' (integer), 'company_name' (character varying (100)), and 'order_count' (bigint). There are 18 rows of data.

```
12 SELECT * FROM view_customer_order_count_year;
```

Data Output Сообщения План выполнения X Notifications

| | customer_id integer | company_name character varying (100) | order_count bigint |
|----|------------------------|---|-----------------------|
| 1 | 652 | АО "ОптМаркет-931" | 2 |
| 2 | 273 | ПАО "Снабжение-359" | 2 |
| 3 | 51 | ЗАО "Торговля-246" | 2 |
| 4 | 839 | ООО "Ритейл-536" | 1 |
| 5 | 539 | ИП "Поставка-779" | 5 |
| 6 | 874 | ПАО "Поставка-881" | 1 |
| 7 | 946 | ЗАО "Продукты-761" | 1 |
| 8 | 176 | ООО "Продукты-419" | 2 |
| 9 | 576 | АО "Торговля-137" | 1 |
| 10 | 292 | ПАО "Снабжение-993" | 1 |
| 11 | 663 | ООО "Снабжение-339" | 2 |
| 12 | 929 | ПАО "Поставка-440" | 2 |
| 13 | 770 | АО "Снабжение-921" | 3 |
| 14 | 271 | ООО "Поставка-430" | 1 |
| 15 | 556 | ЗАО "ОптМаркет-844" | 3 |
| 16 | 638 | ПАО "Торговля-585" | 5 |
| 17 | 791 | АО "Снабжение-118" | 2 |
| 18 | 390 | ИП "Снабжение-212" | 2 |

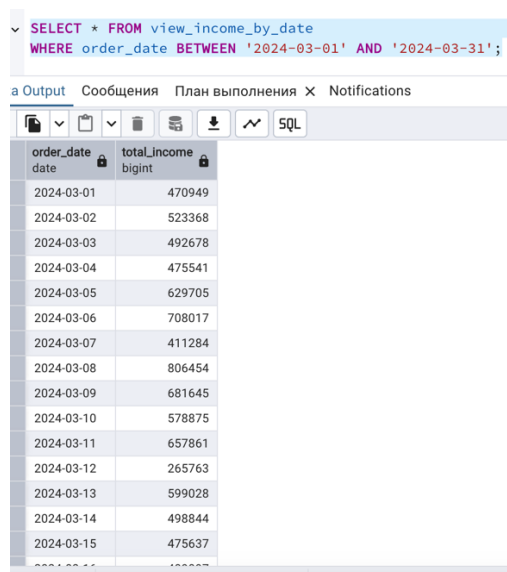
Рисунок 10 – Скриншот просмотра содержимого запроса 1 с результатом

2.Формулировка запроса: доход базы за конкретный период.

```
CREATE VIEW view_income_by_date AS
SELECT
    o.order_date, -- дата, когда был оформлен заказ
    SUM(od.total_price) AS total
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
GROUP BY o.order_date
ORDER BY o.order_date;
SELECT * FROM view_income_by_date
WHERE order_date BETWEEN '2024-03-01' AND '2024-03-31';
```

```
-- 2 представление
-- доход базы за конкретный период.
CREATE VIEW view_income_by_date AS
SELECT
    o.order_date,
    SUM(od.total_price) AS total_income -- сумма по всем строкам заказа за эту дату
FROM orders o
JOIN order_details od ON o.order_id = od.order_id
-- Соединяю таблицу заказов с деталями заказа, чтобы получить доступ к total_price
-- Каждый заказ может содержать несколько строк (товаров), и у каждой строки есть своя стоимость
GROUP BY o.order_date
ORDER BY o.order_date;
```

Рисунок 11 – Скриншот создания представления 2



| order_date | total_income |
|------------|--------------|
| 2024-03-01 | 470949 |
| 2024-03-02 | 523368 |
| 2024-03-03 | 492678 |
| 2024-03-04 | 475541 |
| 2024-03-05 | 629705 |
| 2024-03-06 | 708017 |
| 2024-03-07 | 411284 |
| 2024-03-08 | 806454 |
| 2024-03-09 | 681645 |
| 2024-03-10 | 578875 |
| 2024-03-11 | 657861 |
| 2024-03-12 | 265763 |
| 2024-03-13 | 599028 |
| 2024-03-14 | 498844 |
| 2024-03-15 | 475637 |

Рисунок 12 – Скриншот просмотра содержимого запроса 2 с результатом

2.3 Запросы на модификацию данных

1. Формулировка запроса: INSERT: Добавляем заказ от клиента, у которого больше всего заказов.

```
INSERT INTO orders (customer_id, order_date, delivery_date, payment_status,
processing_status, manager_id, branch_id)
VALUES (
(
SELECT customer_id
FROM orders
GROUP BY customer_id
ORDER BY COUNT(order_id) DESC
LIMIT 1
),
DATE '2024-04-25',
DATE '2024-04-30',
'ожидание',
'новый',
1,
1
);
```

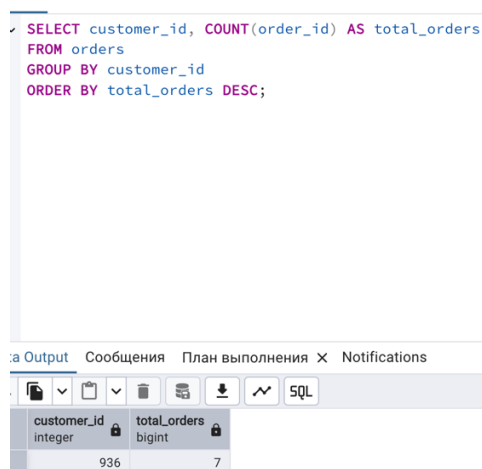


Рисунок 13 – Скриншот до выполнения запроса

```

1  INSERT INTO orders (customer_id, order_date, delivery_date, payment_status, processing_status, manager_id, branch_id)
2  VALUES (
3      -- подзапрос выбирает ID клиента с максимальным количеством заказов
4      (
5          SELECT customer_id
6          FROM orders
7          GROUP BY customer_id
8          ORDER BY COUNT(order_id) DESC
9          LIMIT 1
10     ),
11     DATE '2024-04-25',
12     DATE '2024-04-30',
13     'ожидание',
14     'новый',
15     1,
16     1
17 );
18

```

Data Output Сообщения План выполнения X Notifications

INSERT 0 1

Запрос завершён успешно, время выполнения: 66 мсек.

Рисунок 14 – Скриншот выполнения запроса

```

SELECT customer_id, COUNT(order_id) AS total_orders
FROM orders
GROUP BY customer_id
ORDER BY total_orders DESC;

```

Output Сообщения План выполнения X Notifications

| customer_id | total_orders |
|-------------|--------------|
| 936 | 8 |

Рисунок 15 – Скриншот после выполнения запроса

2. Формулировка запроса: UPDATE: Увеличиваем цену на 10% для товаров, чья цена ниже средней.

```

UPDATE supply_details
SET unit_price = unit_price * 1.10
WHERE unit_price < (
    SELECT AVG(unit_price)

```

);
FROM supply_details

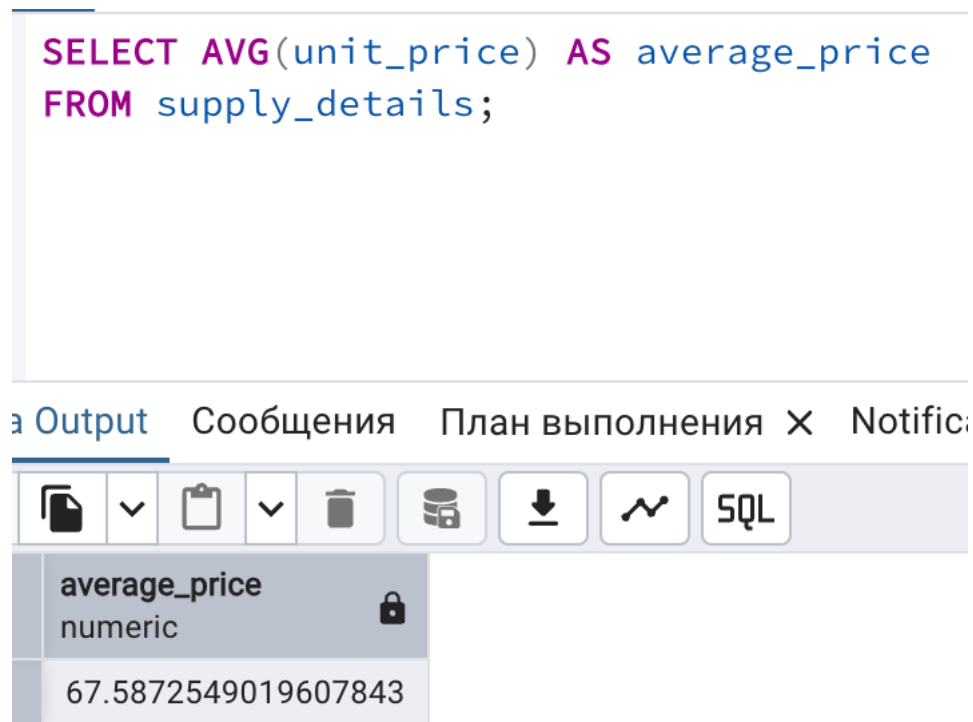
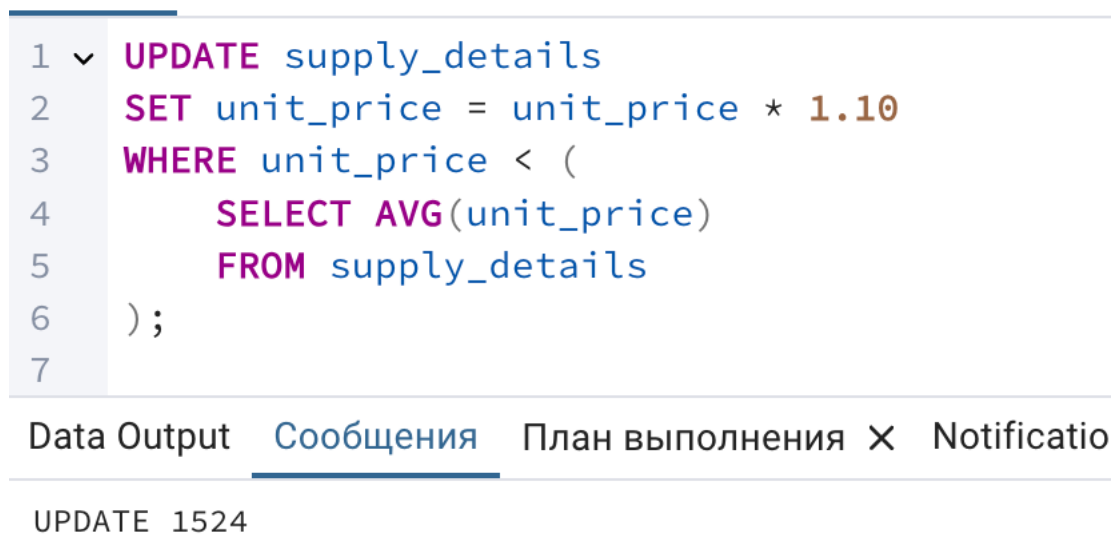


Рисунок 14 – Скриншот до выполнения запроса 2



Запрос завершён успешно, время выполнения: 122 msec.

Рисунок 17 – Скриншот выполнения запроса update

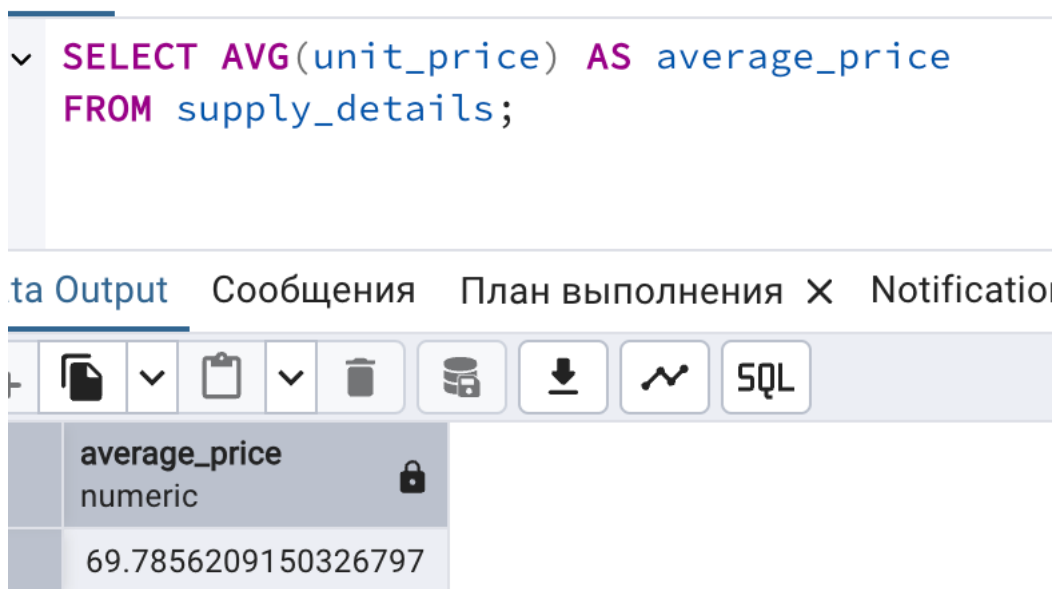


Рисунок 18 – Скриншот после выполнения запроса

3. Формулировка запроса: DELETE: Удаляем поставщиков, которые не поставляют ни одного товара.

```
DELETE FROM supplier
WHERE supplier_id NOT IN (
    SELECT DISTINCT sp.supplier_id
    FROM supply sp
    JOIN supply_details sd ON sp.supply_id = sd.supply_id
);
```

```
SELECT s.supplier_id, s.company_name, COUNT(sd.supply_details_id) AS product_count
FROM supplier s
LEFT JOIN supply sp ON s.supplier_id = sp.supplier_id
LEFT JOIN supply_details sd ON sp.supply_id = sd.supply_id
GROUP BY s.supplier_id, s.company_name
ORDER BY product_count;
```

Output Сообщения План выполнения X Notifications

| supplier_id [PK] integer | company_name character varying (100) | product_count bigint |
|-----------------------------|---|-------------------------|
| 54 | ООО "БезПоставок" | 0 |
| 52 | ООО "ДопПоставщик" | 1 |
| 51 | ООО "ПоставщикВсеТовары" | 10 |

Рисунок 19 – Скриншот до выполнения запроса

```
1 DELETE FROM supplier
2 WHERE supplier_id NOT IN (
3     SELECT DISTINCT sp.supplier_id
4     FROM supply sp
5     JOIN supply_details sd ON sp.supply_id = sd.supply_id
6 );
7
```

Data Output **Сообщения** План выполнения X Notifications

DELETE 1

Запрос завершён успешно, время выполнения: 81 msec.

Рисунок 20 – Скриншот выполнения запроса

```
1 SELECT s.supplier_id, s.company_name, COUNT(sd.supply_details_id) AS product_count
2 FROM supplier s
3 LEFT JOIN supply sp ON s.supplier_id = sp.supplier_id
4 LEFT JOIN supply_details sd ON sp.supply_id = sd.supply_id
5 GROUP BY s.supplier_id, s.company_name
6 ORDER BY product_count;
7
```

Data Output Сообщения План выполнения X Notifications

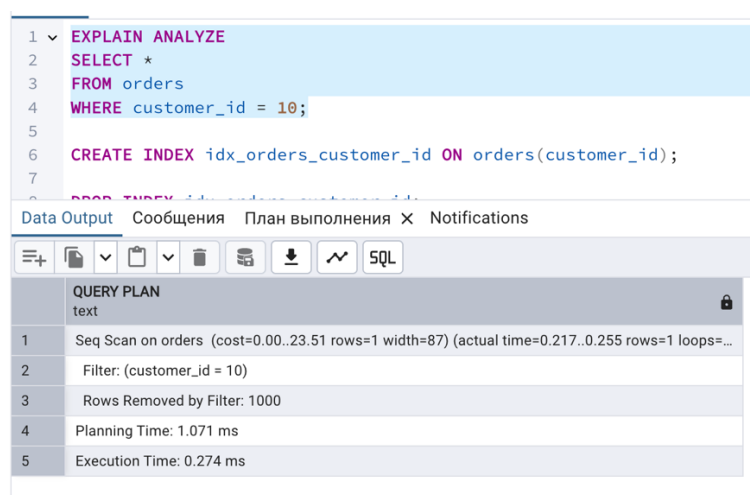
| | supplier_id [PK] integer | company_name character varying (100) | product_count bigint |
|---|-----------------------------|---|-------------------------|
| 1 | 52 | ООО "ДопПоставщик" | 1 |

Рисунок 21 – Скриншот после выполнения запроса

2.4 Создание индексов

1. Формулировка запроса: Найти все заказы по определённому покупателю (customer_id).

```
EXPLAIN ANALYZE
SELECT *
FROM orders
WHERE customer_id = 10;
```



The screenshot shows a database query execution interface. The top part displays the SQL query: `EXPLAIN ANALYZE SELECT * FROM orders WHERE customer_id = 10;` followed by `CREATE INDEX idx_orders_customer_id ON orders(customer_id);`. Below the query, there is a tabbed interface with 'Data Output', 'Сообщения', 'План выполнения', and 'Notifications'. The 'План выполнения' (Execution Plan) tab is selected, showing a table with 5 rows of execution details.

| | QUERY PLAN |
|---|--|
| 1 | Seq Scan on orders (cost=0.00..23.51 rows=1 width=87) (actual time=0.217..0.255 rows=1 loops=... |
| 2 | Filter: (customer_id = 10) |
| 3 | Rows Removed by Filter: 1000 |
| 4 | Planning Time: 1.071 ms |
| 5 | Execution Time: 0.274 ms |

Рисунок 22 – Выполнение запроса без индексов

Создаём индекс на customer_id:

```
CREATE INDEX idx_orders_customer_id ON orders(customer_id);
```

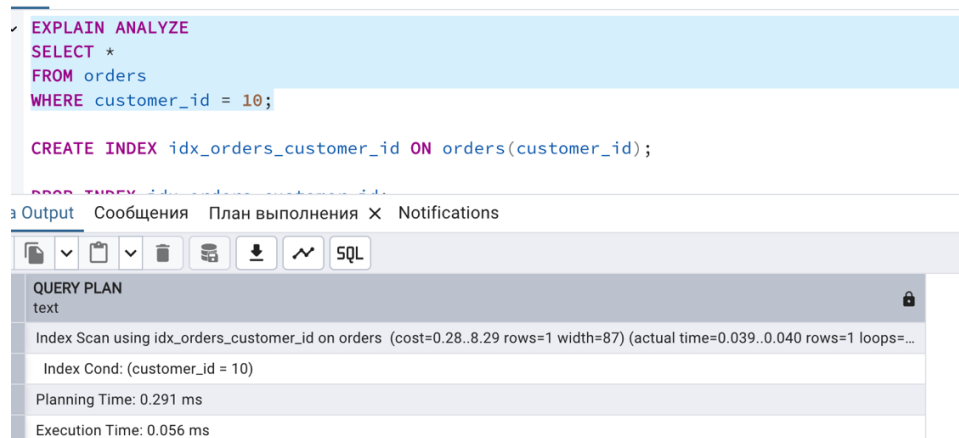


Рисунок 23 – Выполнение запроса с индексами

Удаляем индекс:

DROP INDEX idx_orders_customer_id;

2. Формулировка запроса: Найти заказы по дате заказа (order_date).

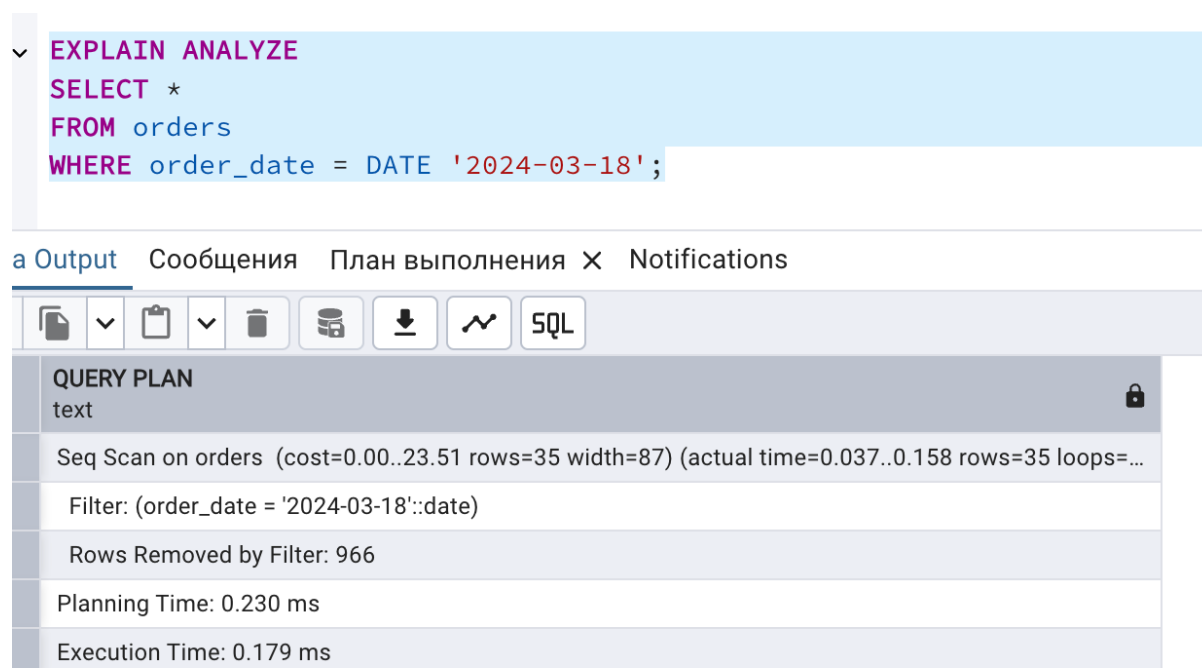


Рисунок 24 - Выполнение запроса без индексов

```
EXPLAIN ANALYZE
SELECT *
FROM orders
WHERE order_date = DATE '2024-03-18';

CREATE INDEX idx_orders_order_date ON orders(order_date);
```

Output Сообщения План выполнения ✕ Notifications

QUERY PLAN

text

Bitmap Heap Scan on orders (cost=4.42..15.86 rows=35 width=87) (actual time=0.026..0.043 rows=35 loops=1)

 Recheck Cond: (order_date = '2024-03-18'::date)

 Heap Blocks: exact=9

 -> Bitmap Index Scan on idx_orders_order_date (cost=0.00..4.41 rows=35 width=0) (actual time=0.016..0.017 rows=35 loop...)

 Index Cond: (order_date = '2024-03-18'::date)

Planning Time: 0.069 ms

Execution Time: 0.061 ms

Рисунок 25 – Выполнение запроса с индексами

ВЫВОД

В ходе выполнения лабораторной работы были изучены и практически освоены различные операции работы с базой данных PostgreSQL.

Были выполнены следующие задачи:

- Созданы таблицы базы данных с необходимыми ограничениями (PRIMARY KEY, FOREIGN KEY, CHECK, UNIQUE).
- Реализовано наполнение таблиц данными, обеспечивающими связность между объектами базы данных.
- Разработаны и выполнены сложные запросы на выборку данных с использованием подзапросов, группировок, агрегатных функций и условий фильтрации.
- Созданы представления (VIEW) для упрощения работы с часто используемыми запросами и представления агрегированных данных.
- Реализованы запросы на модификацию данных (INSERT, UPDATE, DELETE) с использованием вложенных подзапросов, что позволило повысить сложность и осмысленность выполнения операций.
- Проведено создание индексов на ключевых полях таблиц для оптимизации скорости выполнения запросов. Выполнено сравнение планов выполнения запросов до и после создания индексов с использованием EXPLAIN ANALYZE.
- Проанализированы планы выполнения, время работы запросов без индексов и с индексами, сделаны выводы об эффективности использования индексов.

В результате выполнения лабораторной работы были закреплены практические навыки:

- проектирования структуры базы данных,
- оптимизации производительности запросов,
- грамотной организации выборки и модификаций данных,
- использования представлений для упрощения работы с данными.

Поставленные цели лабораторной работы были полностью достигнуты.