САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2
по курсу «Проектирование и реализация баз данных»
Тема: Запросы на выборку и модификацию данных, представления и индексы в PostgreSQL

Выполнил:
Седельников П.В.
К32401

Проверила:
Говорова М.М.

Санкт-Петербург
2023 г.

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

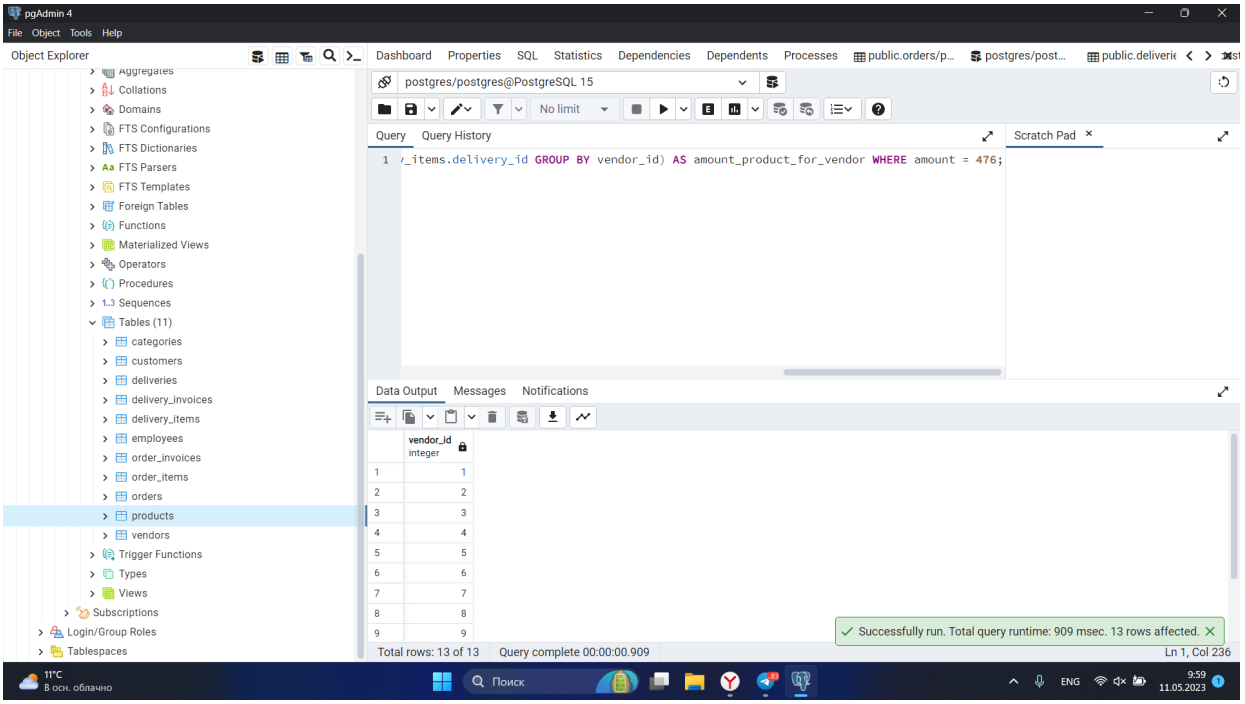
Практическое задание:

1. Создать запросы и представления на выборку данных к базе данных PostgreSQL (согласно индивидуальному заданию, 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. Запросы

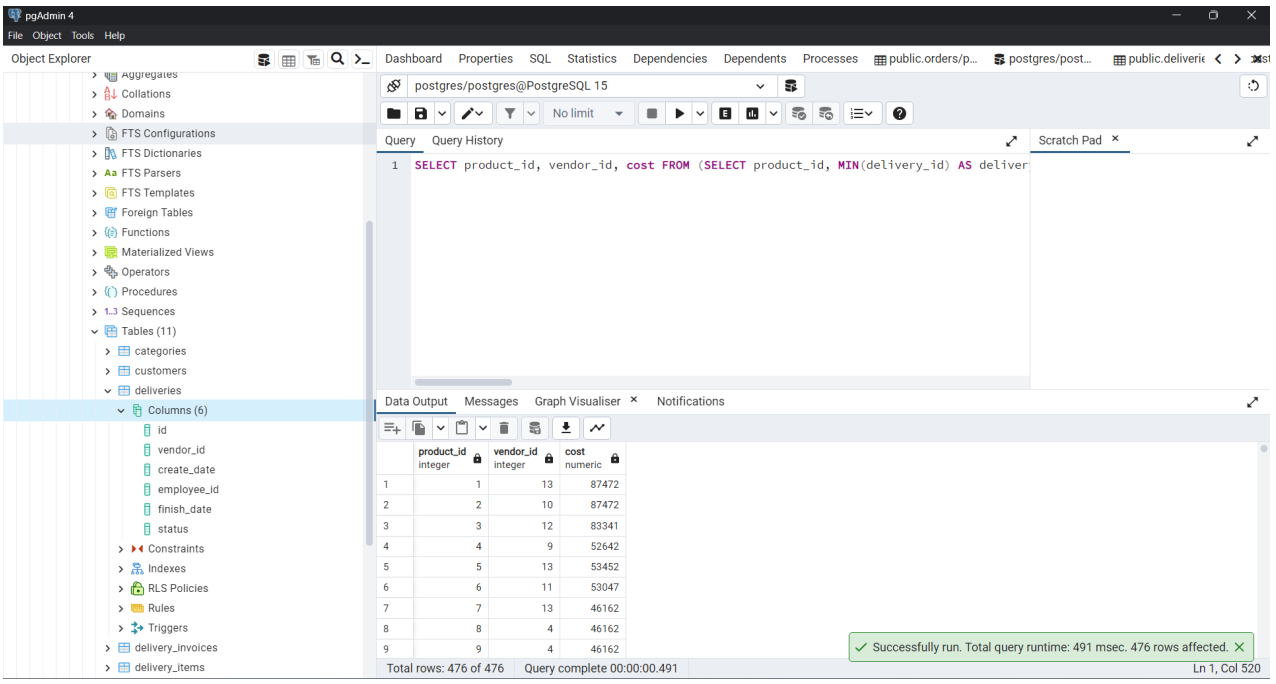
1.1. Вывести список поставщиков, которые поставляют все товары.

SELECT vendor_id FROM (SELECT vendor_id, COUNT(DISTINCT product_id) AS amount FROM public.deliveries JOIN delivery_items ON deliveries.id = delivery_items.delivery_id GROUP BY vendor_id) AS amount_product_for_vendor WHERE amount = (SELECT COUNT(*) FROM products);



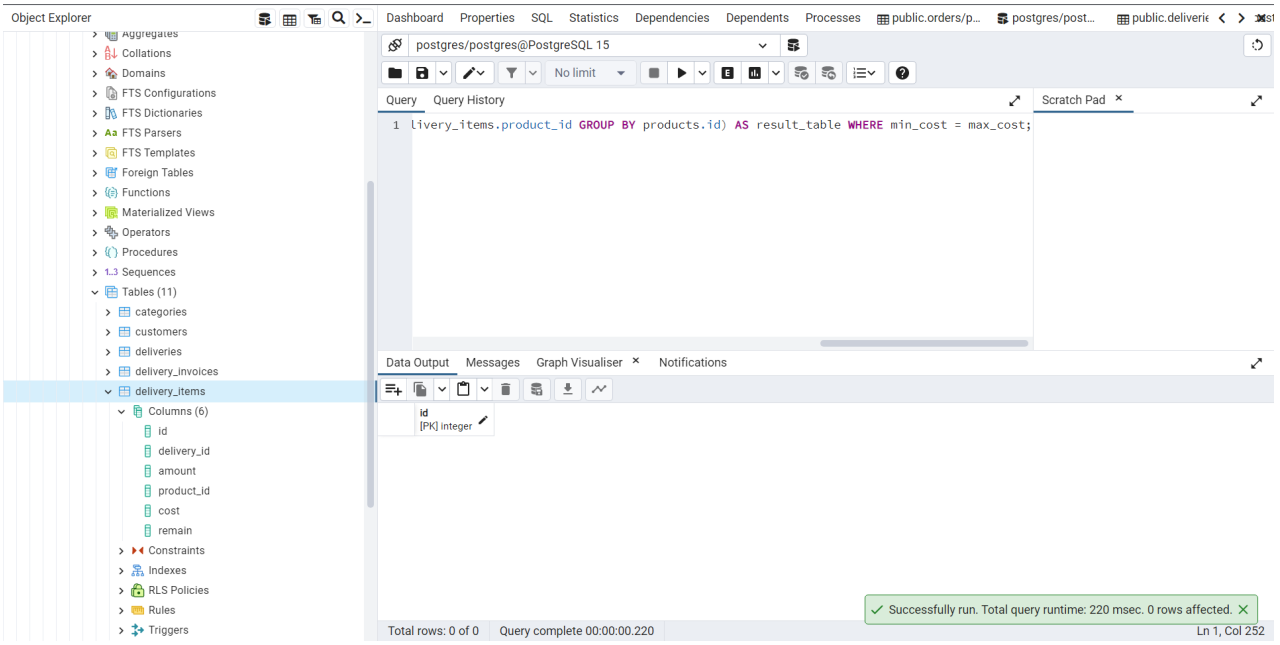
1.2. Определить поставщика, который поставляет каждый из товаров по самой низкой цене.

SELECT product_id, vendor_id, cost FROM (SELECT product_id, MIN(delivery_id) AS delivery_id, MIN(cost) AS cost FROM (SELECT DISTINCT product_id, delivery_id, cost FROM delivery_items JOIN (SELECT products.id, MIN(delivery_items.cost) AS min_cost FROM public.products JOIN public.delivery_items ON products.id = delivery_items.product_id GROUP BY products.id) AS min_cost_table ON product_id = min_cost_table.id WHERE cost = min_cost) AS cost_table GROUP BY product_id) AS final_table JOIN deliveries ON id = delivery_id;



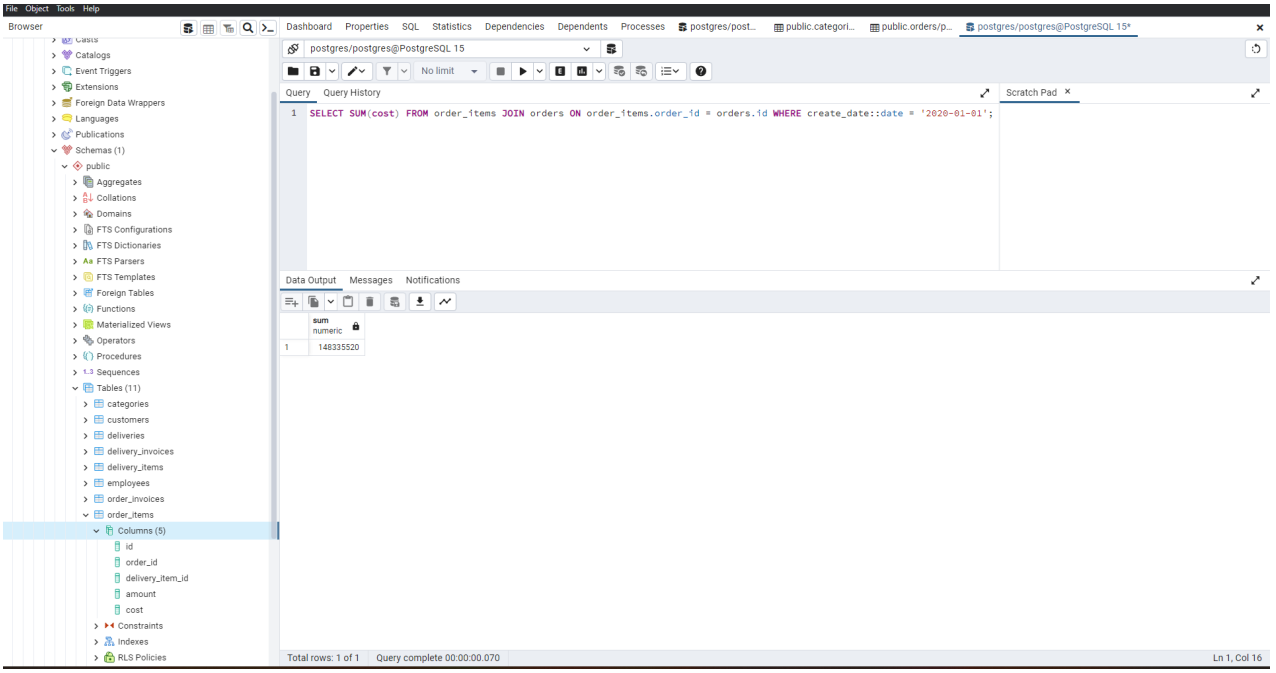
1.3. Вывести названия товаров, цены на которые у всех поставщиков одинаковы.

SELECT id FROM (SELECT products.id, MIN(delivery_items.cost) AS min_cost, MAX(delivery_items.cost) AS max_cost FROM products JOIN delivery_items ON products.id = delivery_items.product_id GROUP BY products.id) AS result_table WHERE min_cost = max_cost;



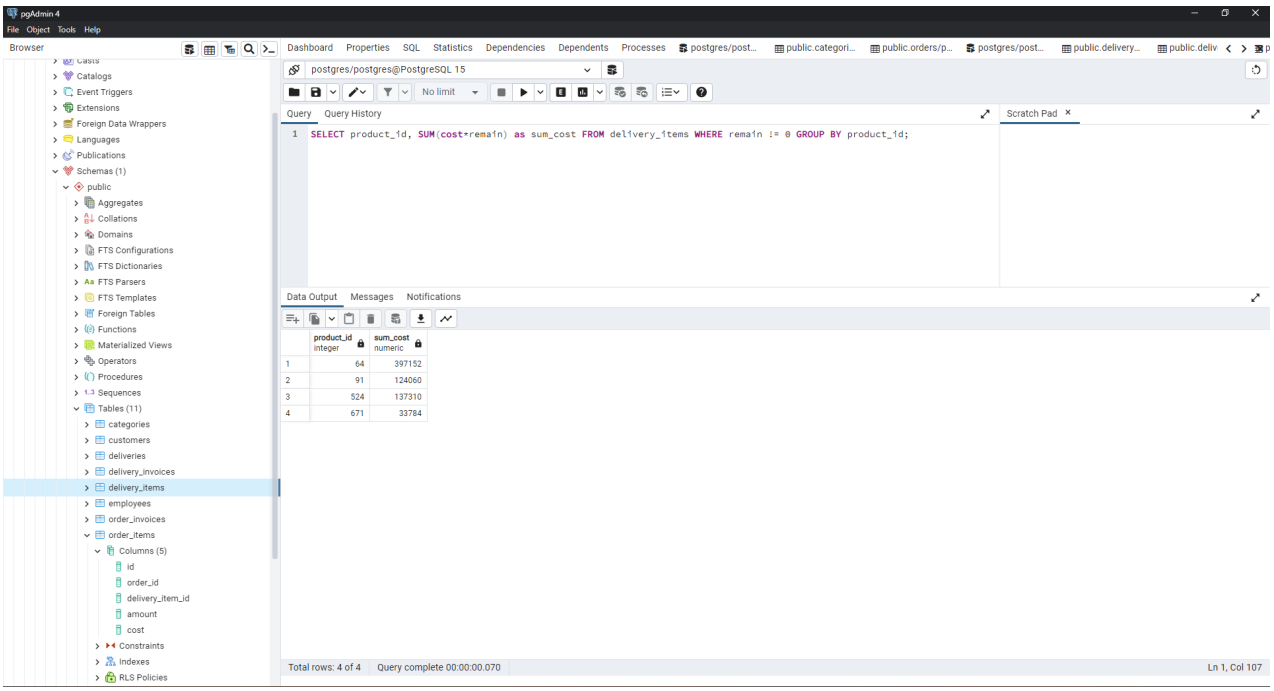
1.4. Чему равен общий суточный доход оптового склада за прошедший день?

SELECT SUM(cost) FROM order_items JOIN orders ON order_items.order_id = orders.id WHERE create_date::date = CURRENT_DATE - 1;



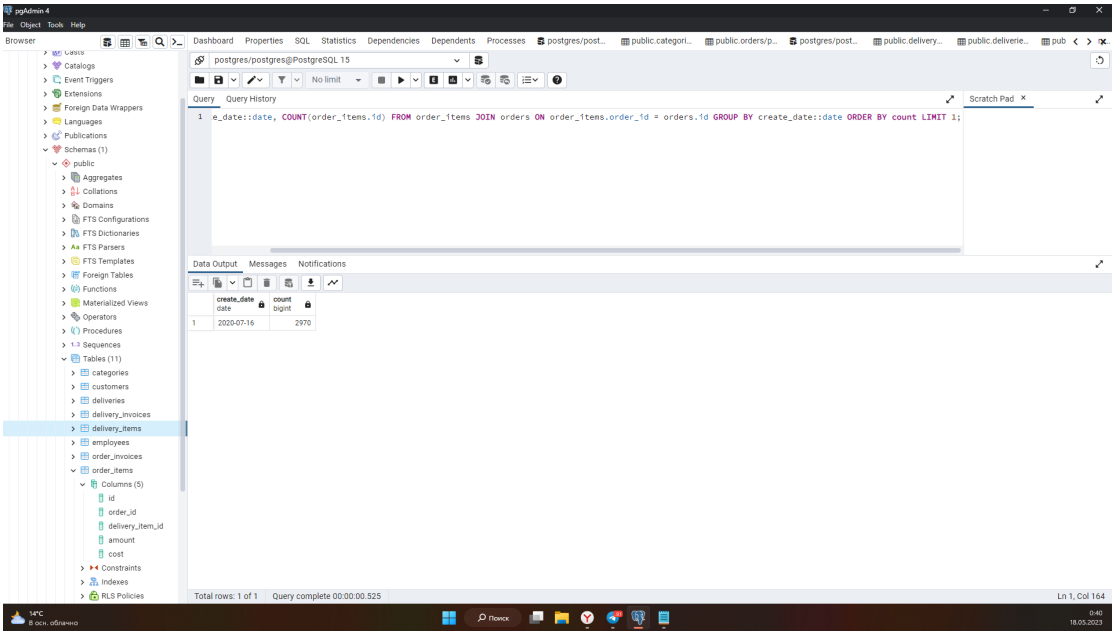
1.3. Вычислить общую стоимость каждого вида товара, находящегося на базе.

SELECT product_id, SUM(cost*remain) as sum_cost FROM delivery_items WHERE remain != 0 GROUP BY product_id;



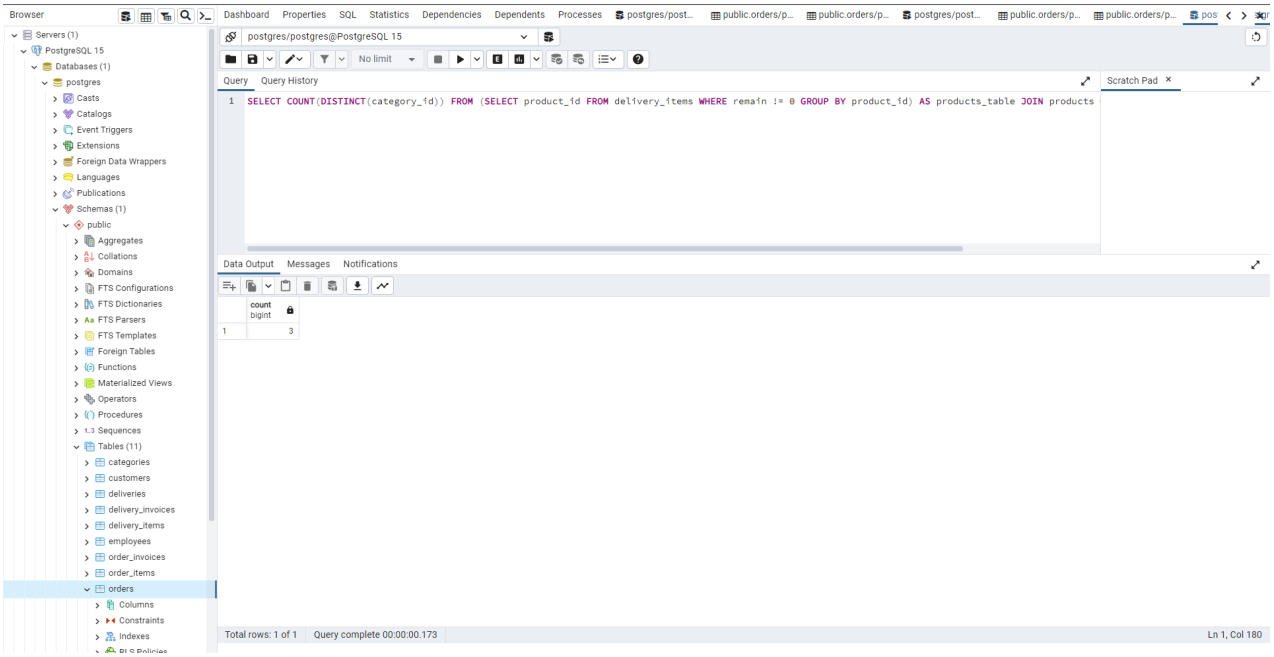
1.4. В какой день было вывезено минимальное количество товара?

SELECT create_date, count FROM (SELECT create_date::date, COUNT(order_items.id) FROM order_items JOIN orders ON order_items.order_id = orders.id GROUP BY create_date::date) AS result_table WHERE count = (SELECT COUNT(order_items.id) FROM order_items JOIN orders ON order_items.order_id = orders.id GROUP BY create_date::date ORDER BY count LIMIT 1);



1.5. Сколько различных видов товара имеется на базе?

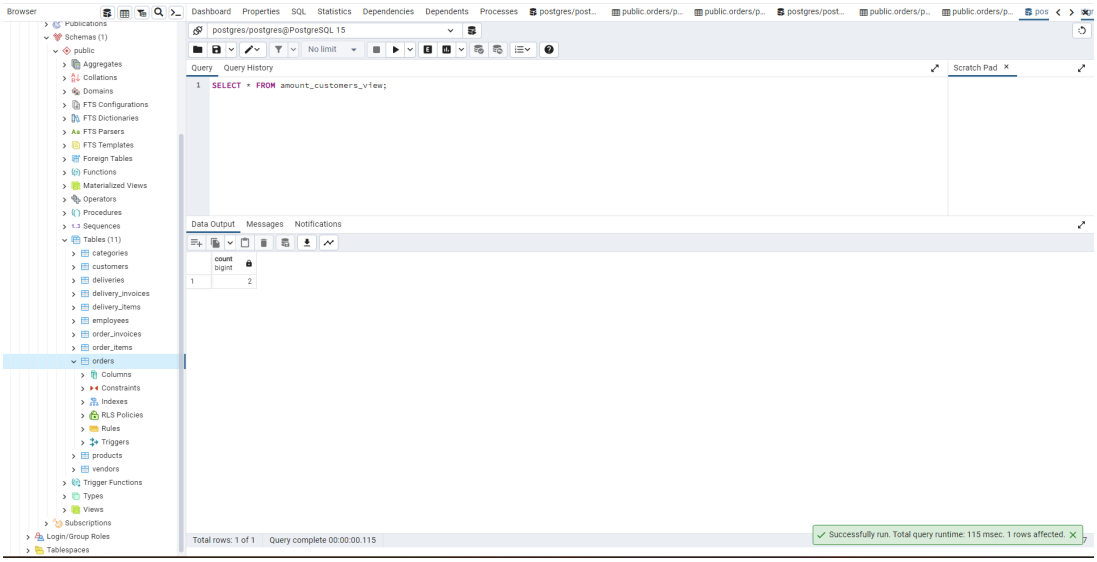
```
SELECT COUNT(DISTINCT(category_id)) FROM (SELECT product_id
FROM delivery_items WHERE remain != 0 GROUP BY product_id) AS
products_table JOIN products ON products.id = product_id;
```



2. Представления

2.1. Количество заказов фирм-покупателей за прошедший год;

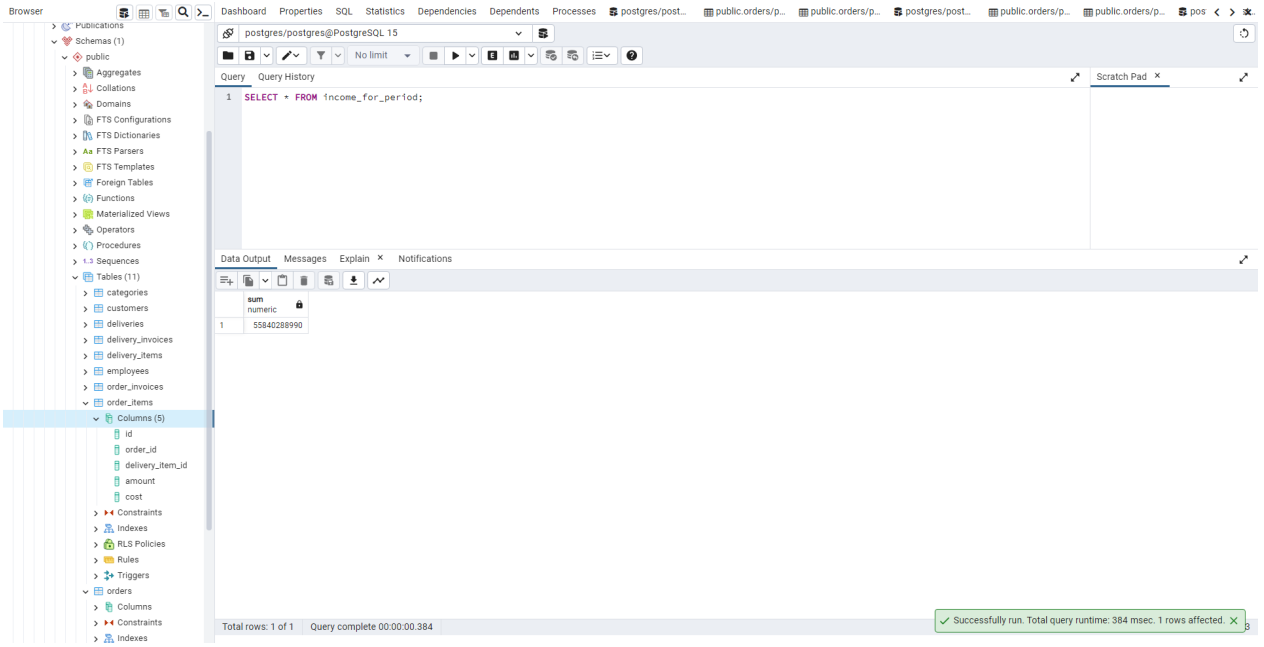
```
CREATE VIEW amount_customers_view AS SELECT COUNT(DISTINCT(customer_id))
FROM orders WHERE create_date::date > CURRENT_DATE - INTERVAL '1 YEAR';
SELECT * FROM amount_customers_view ;
```



2.2. Доход базы за конкретный период.

CREATE VIEW income_for_period AS SELECT SUM(cost*amount) FROM order_items JOIN orders ON order_items.order_id = orders.id WHERE create_date::date >= '2020-01-01' AND create_date::date <= '2020-12-31';

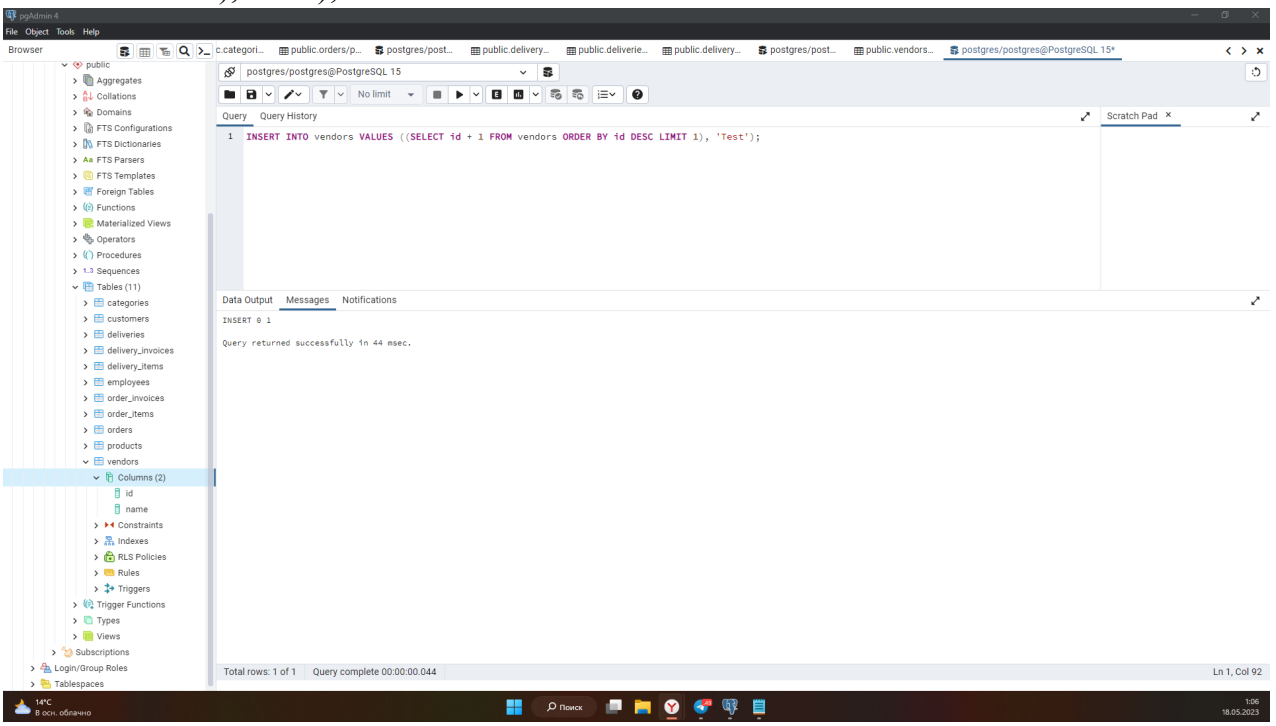
SELECT * FROM income_for_period ;



3. Модификация данных

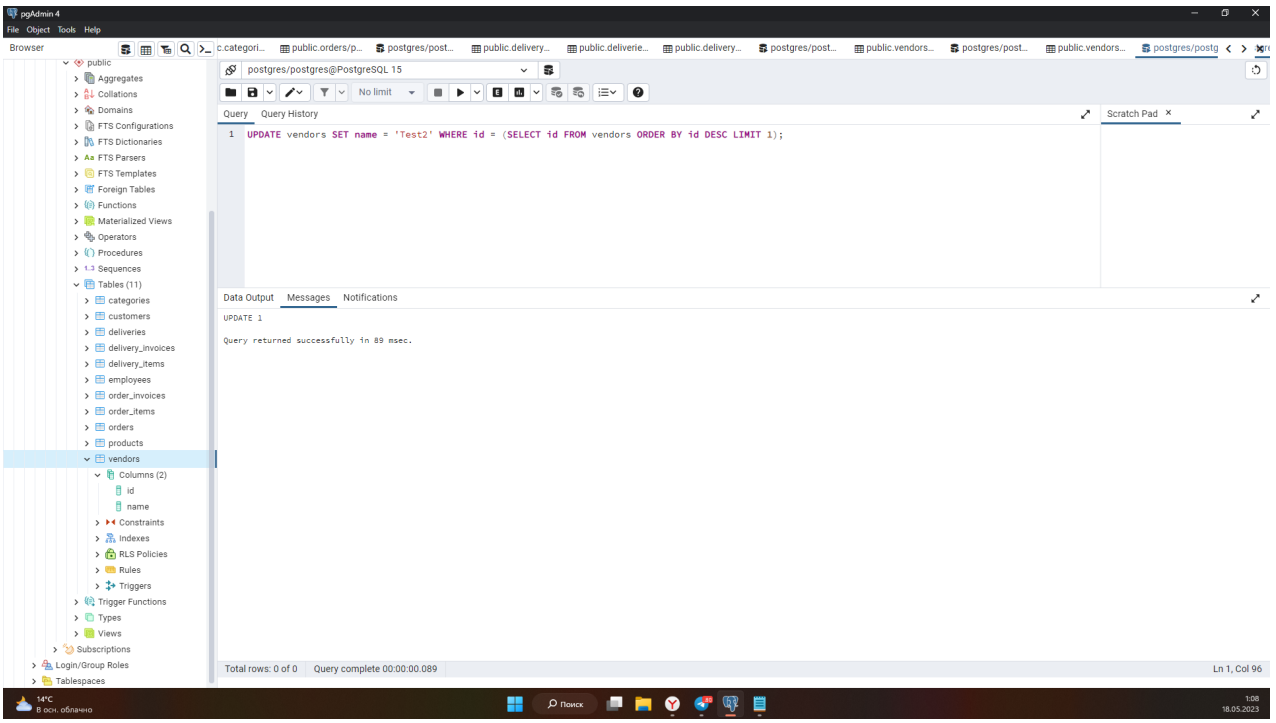
3.1. INSERT

INSERT INTO vendors VALUES ((SELECT id + 1 FROM vendors ORDER BY id DESC LIMIT 1), 'Test');



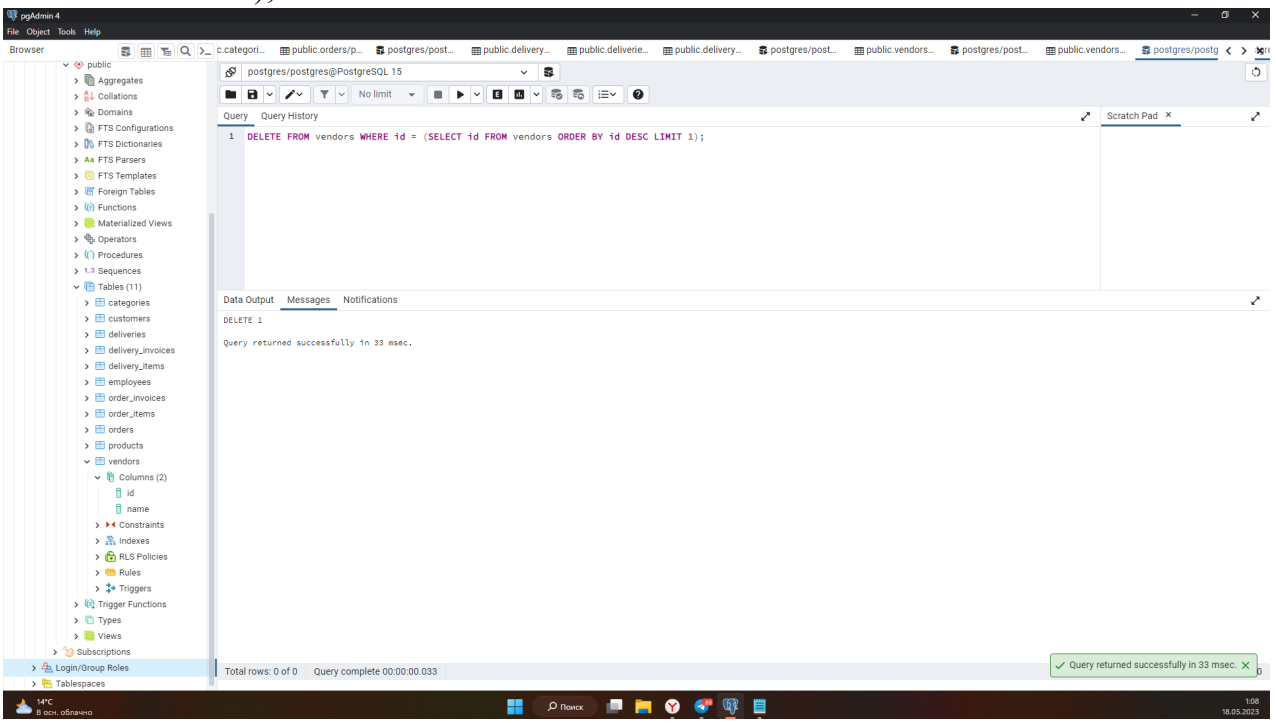
3.2. UPDATE

UPDATE vendors SET name = 'Test2' WHERE id = (SELECT id FROM vendors ORDER BY id DESC LIMIT 1);



3.3. DELETE

DELETE FROM vendors WHERE id = (SELECT id FROM vendors ORDER BY id DESC LIMIT 1);



4. Сравнение запросов

4.1. Первый запрос

SELECT create_date, count FROM (SELECT create_date::date,

COUNT(order_items.id) FROM order_items JOIN orders ON

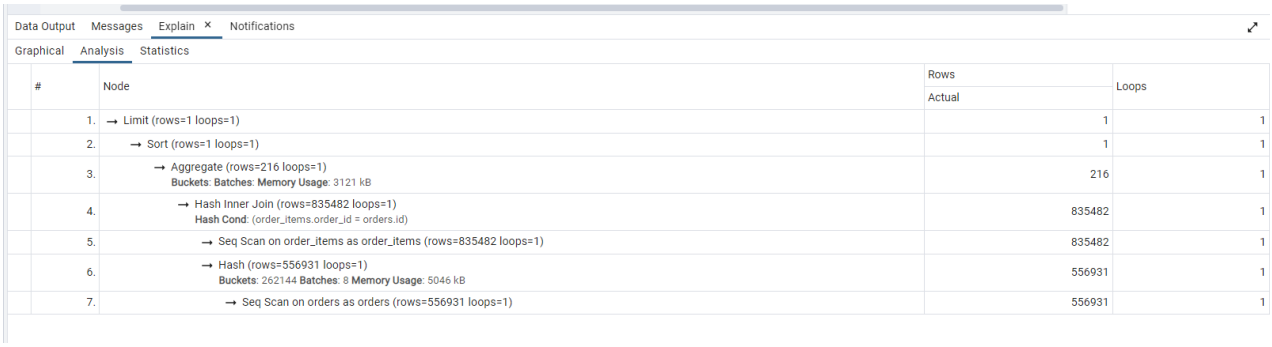
order_items.order_id = orders.id GROUP BY create_date::date) AS

result_table WHERE count = (SELECT COUNT(order_items.id) FROM order_items JOIN orders ON order_items.order_id = orders.id GROUP BY create_date::date ORDER BY count LIMIT 1);

Индексы:

```
CREATE INDEX order_id_index
    ON public.order_items USING btree
    (order_id ASC NULLS LAST);
CREATE INDEX create_date_index
    ON public.orders USING btree
    (create_date ASC NULLS LAST);
```

Без индексов: 2200 мс.

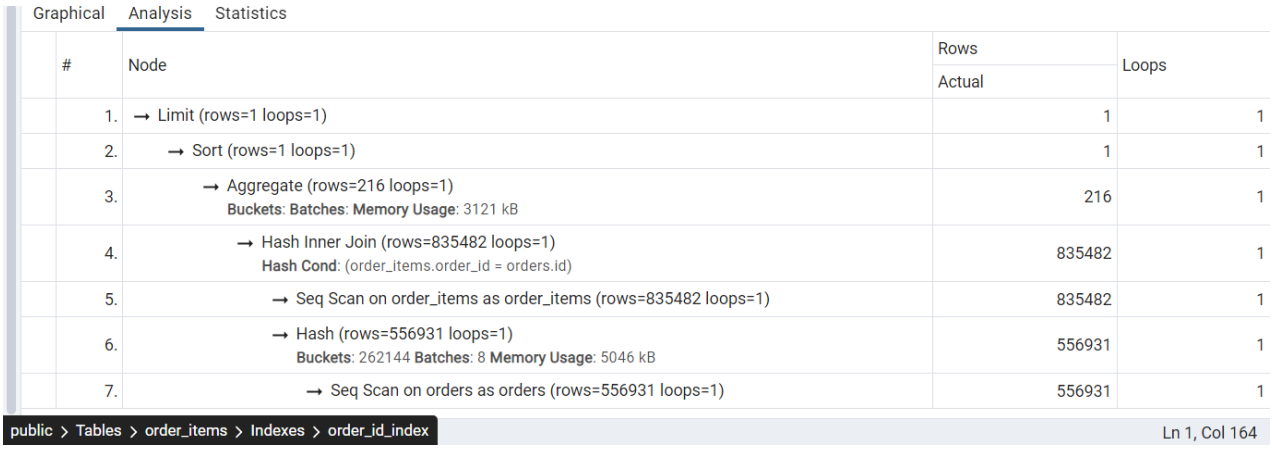


Data Output | Messages | Explain | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Limit (rows=1 loops=1)	1	1
2.	→ Sort (rows=1 loops=1)	1	1
3.	→ Aggregate (rows=216 loops=1) **Buckets**: **Batches**: **Memory Usage**: 3121 kB	216	1
4.	→ Hash Inner Join (rows=835482 loops=1) **Hash Cond**: (order_items.order_id = orders.id)	835482	1
5.	→ Seq Scan on order_items as order_items (rows=835482 loops=1)	835482	1
6.	→ Hash (rows=556931 loops=1) **Buckets**: 262144 **Batches**: 8 **Memory Usage**: 5046 kB	556931	1
7.	→ Seq Scan on orders as orders (rows=556931 loops=1)	556931	1

С индексом: 794 мс.



Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Limit (rows=1 loops=1)	1	1
2.	→ Sort (rows=1 loops=1)	1	1
3.	→ Aggregate (rows=216 loops=1) **Buckets**: **Batches**: **Memory Usage**: 3121 kB	216	1
4.	→ Hash Inner Join (rows=835482 loops=1) **Hash Cond**: (order_items.order_id = orders.id)	835482	1
5.	→ Seq Scan on order_items as order_items (rows=835482 loops=1)	835482	1
6.	→ Hash (rows=556931 loops=1) **Buckets**: 262144 **Batches**: 8 **Memory Usage**: 5046 kB	556931	1
7.	→ Seq Scan on orders as orders (rows=556931 loops=1)	556931	1

public > Tables > order_items > Indexes > order_id_index Ln 1, Col 164

4.2. Второй запрос

SELECT product_id, SUM(cost*remain) as sum_cost FROM delivery_items WHERE remain != 0 GROUP BY product_id;

Индексы:

```
CREATE INDEX remain_index
    ON public.delivery_items USING btree
    (remain ASC NULLS LAST);
```

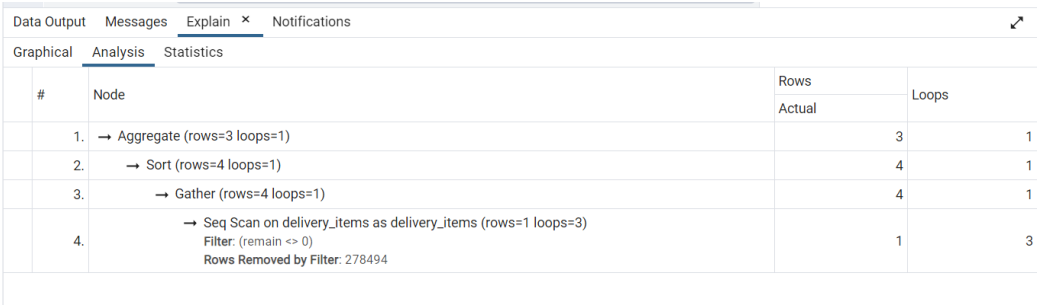
Без индексов: 543 мс.

Data Output | Messages | Explain | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Aggregate (rows=4 loops=1)	4	1
2.	→ Sort (rows=5 loops=1)	5	1
3.	→ Gather (rows=5 loops=1)	5	1
4.	→ Seq Scan on delivery_items as delivery_items (rows=2 loops=3) **Filter**: (remain <> 0) **Rows Removed by Filter**: 278494	2	3

С индексом: 180 мс.



Data Output | Messages | Explain | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Aggregate (rows=3 loops=1)	3	1
2.	→ Sort (rows=4 loops=1)	4	1
3.	→ Gather (rows=4 loops=1)	4	1
4.	→ Seq Scan on delivery_items as delivery_items (rows=1 loops=3) **Filter**: (remain <> 0) **Rows Removed by Filter**: 278494	1	3

4.3. Третий запрос

SELECT SUM(cost) FROM order_items JOIN orders ON order_items.order_id = orders.id WHERE create_date::date = CURRENT_DATE - 1;

Индексы:

```
CREATE INDEX order_id_index
    ON public.order_items USING btree
    (order_id ASC NULLS LAST);
CREATE INDEX create_date_index
    ON public.orders USING btree
    (create_date ASC NULLS LAST);
```

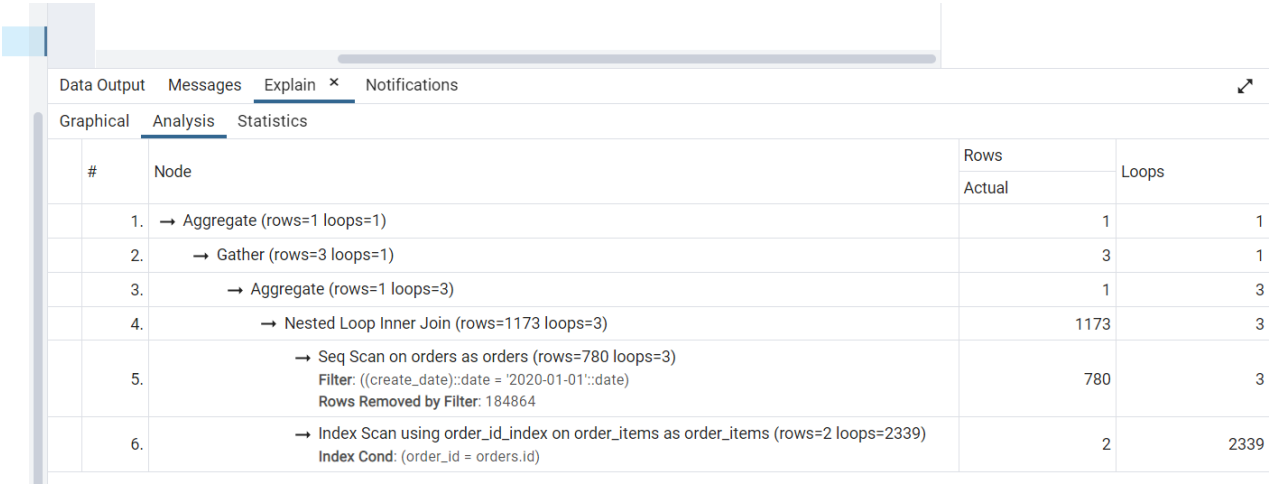
Без индексов: 331 мс.

Data Output | Messages | Explain × | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Aggregate (rows=1 loops=1)	1	1
2.	→ Gather (rows=3 loops=1)	3	1
3.	→ Aggregate (rows=1 loops=3)	1	3
4.	→ Nested Loop Inner Join (rows=1173 loops=3)	1173	3
5.	→ Seq Scan on orders as orders (rows=780 loops=3) **Filter**: ((create_date)::date = '2020-01-01'::date) **Rows Removed by Filter**: 184864	780	3
6.	→ Index Scan using fki_orders_fkey on order_items as order_items (rows=2 loops=2339) **Index Cond**: (order_id = orders.id)	2	2339

С индексом: 255 мс.



Data Output | Messages | Explain × | Notifications

Graphical | Analysis | Statistics

#	Node	Rows Actual	Loops
1.	→ Aggregate (rows=1 loops=1)	1	1
2.	→ Gather (rows=3 loops=1)	3	1
3.	→ Aggregate (rows=1 loops=3)	1	3
4.	→ Nested Loop Inner Join (rows=1173 loops=3)	1173	3
5.	→ Seq Scan on orders as orders (rows=780 loops=3) **Filter**: ((create_date)::date = '2020-01-01'::date) **Rows Removed by Filter**: 184864	780	3
6.	→ Index Scan using order_id_index on order_items as order_items (rows=2 loops=2339) **Index Cond**: (order_id = orders.id)	2	2339

Вывод: овладел практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.