

**Министерство науки и высшего образования Российской Федерации**  
**федеральное государственное автономное образовательное учреждение**  
**высшего образования**  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

**по лабораторной работе № 3**

**«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ, ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В  
POSTGRESQL»**

**по дисциплине «Проектирование и реализация баз данных»**

Автор: Бускина Алия

Факультет: ИКТ

Группа: К32421

Преподаватель: Говорова М.М.

Санкт-Петербург, 2023

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

**Задание 2.** Создайте запросы:

- Вывести список поставщиков, которые поставляют все товары.
  - Определить поставщика, который поставляет каждый из товаров по самой низкой цене.
  - Вывести названия товаров, цены на которые у всех поставщиков одинаковы.
  - Чему равен общий суточный доход оптового склада за прошедший день?
  - Вычислить общую стоимость каждого вида товара, находящегося на базе.
  - В какой день было вывезено минимальное количество товара?
  - Сколько различных видов товара имеется на базе?
- Вывести список поставщиков, которые поставляют все товары.

```
SELECT p.customer_id
FROM product p
GROUP BY p.customer_id
HAVING COUNT(DISTINCT p.product_name) = (SELECT COUNT(DISTINCT
product_name) FROM product);
```

The screenshot shows a PostgreSQL query editor interface. The 'Query' tab is active, displaying the following SQL query:

```
1 SELECT p.customer_id
2 FROM product p
3 GROUP BY p.customer_id
4 HAVING COUNT(DISTINCT p.product_name) = (SELECT COUNT(DISTINCT product_name) FROM product);
```

The 'Data Output' tab is also active, showing the results of the query. The results are displayed in a table with two columns: 'customer\_id' and 'customer\_id' (the second column is likely a duplicate of the first). The results are as follows:

customer_id	customer_id
customer1	customer1
customer10	customer10
customer3	customer3
customer5	customer5

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

```
SELECT p.product_name, c.customer_id, MIN(p.cost_ed) as min_cost
FROM product p
JOIN customer c ON p.customer_id = c.customer_id
GROUP BY p.product_name, c.customer_id
HAVING MIN(p.cost_ed) = (
    SELECT MIN(cost_ed)
    FROM product p2
    WHERE p2.product_name = p.product_name
);
```

Query

Query History

```
1 SELECT p.product_name, c.customer_id, MIN(p.cost_ed) as min_cost
2 FROM product p
3 JOIN customer c ON p.customer_id = c.customer_id
4 GROUP BY p.product_name, c.customer_id
5 HAVING MIN(p.cost_ed) = (
6     SELECT MIN(cost_ed)
7     FROM product p2
8     WHERE p2.product_name = p.product_name
9 );
```

Data Output

Messages

Notifications

	product_name character varying (100)	customer_id character varying (10)	min_cost integer
1	orange	customer5	126
2	melon	customer6	100
3	compote	customer1	156
4	water melon	customer1	160
5	compote	customer3	156
6	juice	customer1	150
7	apple	customer1	100
8	juice	customer3	150
9	grape	customer7	140
10	pear	customer1	100

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

```

SELECT p.product_name
FROM product p
JOIN customer c ON p.customer_id = c.customer_id
WHERE p.cost_ed IN (
    SELECT p2.cost_ed
    FROM product p2
    JOIN customer c2 ON p2.customer_id = c2.customer_id
    WHERE p2.product_name = p.product_name
    GROUP BY p2.cost_ed
    HAVING COUNT(DISTINCT c2.customer_id) = (
        SELECT COUNT(*) FROM customer
    )
)
GROUP BY p.product_name
HAVING COUNT(DISTINCT c.customer_id) = (
    SELECT COUNT(*) FROM customer
);

```

Query Query History

```
1 SELECT p.product_name
2 FROM product p
3 JOIN customer c ON p.customer_id = c.customer_id
4 WHERE p.cost_ed IN (
5     SELECT p2.cost_ed
6     FROM product p2
7     JOIN customer c2 ON p2.customer_id = c2.customer_id
8     WHERE p2.product_name = p.product_name
9     GROUP BY p2.cost_ed
10    HAVING COUNT(DISTINCT c2.customer_id) = (
11        SELECT COUNT(*) FROM customer
12    )
13 )
14 GROUP BY p.product_name
15 HAVING COUNT(DISTINCT c.customer_id) = (
16     SELECT COUNT(*) FROM customer
17 );
```

Data Output Messages Notifications



	product_name character varying (100) 🔒
1	dragon_fruit
2	feijoa
3	stawberry

- Чему равен общий суточный доход оптового склада за прошедший день?

```
SELECT SUM(sum_of_order)*0.05 AS daily_income
```

```
FROM order_invoice
```

```
WHERE payment_date = '2023-03-30';
```

Query Query History

```
1 SELECT SUM(sum_of_order)*0.05 AS daily_income
2 FROM order_invoice
3 WHERE payment_date = '2023-03-30';
```

Data Output Messages Notifications



	daily_income numeric 🔒
1	3603.05

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

```
SELECT p.product_name, SUM(p.cost_ed * s.amount_of_products) AS
total_cost
FROM product p
JOIN purchase_composition s ON p.product_id = s.product_id
GROUP BY p.product_name;
```

Query		Query History
<pre>1 SELECT p.product_name, SUM(p.cost_ed * s.amount_of_products) AS total_cost 2 FROM product p 3 JOIN purchase_composition s ON p.product_id = s.product_id 4 GROUP BY p.product_name;</pre>		
Data Output		Messages
		Notifications
	product_name character varying (100)	total_cost bigint
1	water melon	17275
2	dragon_fruit	161840
3	melon	7040
4	grape	11000
5	feijoa	37800
6	compote	2242
7	orange	21496
8	apple	14650
9	juice	7789
10	stawberry	28050
11	pear	8050

- В какой день было вывезено минимальное количество товара?

```
SELECT orders.order_date, SUM(order_composition.amount_of_product) AS total_quantity
FROM orders
JOIN order_composition ON orders.order_id = order_composition.order_id
GROUP BY orders.order_date
HAVING SUM(order_composition.amount_of_product) = (
  SELECT MIN(total_quantity)
  FROM (
    SELECT SUM(order_composition.amount_of_product) AS total_quantity
    FROM orders
    JOIN order_composition ON orders.order_id = order_composition.order_id
    GROUP BY orders.order_date
  ) subquery
)
```

Query

Query History

```
1 SELECT orders.order_date, SUM(order_composition.amount_of_product) AS total_quantity
2 FROM orders
3 JOIN order_composition ON orders.order_id = order_composition.order_id
4 GROUP BY orders.order_date
5 HAVING SUM(order_composition.amount_of_product) = (
6     SELECT MIN(total_quantity)
7     FROM (
8         SELECT SUM(order_composition.amount_of_product) AS total_quantity
9         FROM orders
10        JOIN order_composition ON orders.order_id = order_composition.order_id
11        GROUP BY orders.order_date
12    ) subquery
13 )
```

Data Output

Messages

Notifications

	order_date date	total_quantity bigint
1	2023-03-28	178

- Сколько различных видов товара имеется на базе?

```

SELECT COUNT(DISTINCT product_name) AS total_products
FROM product;

```

Query	Query History				
<pre> 1 SELECT COUNT(DISTINCT product_name) AS total_products 2 FROM product; 3 4 5 </pre>					
Data Output	Messages				
<div> <div> <div>+</div> <div>▼</div> <div>📄</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div> <table> <tr> <th></th><th>total_products bigint</th></tr> <tr> <td>1</td><td>11</td></tr> </table> </div>		total_products bigint	1	11	
	total_products bigint				
1	11				

### Задание 3. Создайте представления:

- количество заказов фирм-покупателей за прошедший год;
- доход базы за конкретный период.

- количество заказов фирм-покупателей за прошедший год;  
CREATE VIEW orders\_per\_buyer\_last\_year AS  
SELECT buyer\_name, COUNT(orders.buyer\_id) as order\_count  
FROM orders  
LEFT JOIN buyer ON orders.buyer\_id = buyer.buyer\_id  
WHERE EXTRACT(year FROM orders.order\_date) = EXTRACT(year FROM NOW()) - 1  
GROUP BY buyer\_name;

Query Query History

```
1 CREATE VIEW orders_per_buyer_last_year AS
2 SELECT buyer_name, COUNT(orders.buyer_id) as order_count
3 FROM orders
4 LEFT JOIN buyer ON orders.buyer_id = buyer.buyer_id
5 WHERE EXTRACT(year FROM orders.order_date) = EXTRACT(year FROM NOW()) - 1
6 GROUP BY buyer_name;
7
```

Data Output Messages Notifications

CREATE VIEW











Query returned successfully in 91 msec.

SELECT \* FROM orders\_per\_buyer\_last\_year;

Query Query History

```
1 SELECT * FROM orders_per_buyer_last_year;
2
```

Data Output Messages Notifications

       			
	buyer_name character varying (100) 	order_count bigint 	
1	company_10	1	
2	company_19	1	

- доход базы за конкретный период  
CREATE VIEW revenue AS  
SELECT  
SUM(order\_invoice.sum\_of\_order) \* 0.05 as revenue\_amount  
FROM  
order\_invoice  
WHERE  
order\_invoice.payment\_date BETWEEN '2022-01-01' AND '2023-12-31';

Query Query History

```
1 CREATE VIEW revenue AS
2 SELECT
3     SUM(order_invoice.sum_of_order) * 0.05 as revenue_amount
4 FROM
5     order_invoice
6 WHERE
7     order_invoice.payment_date BETWEEN '2022-01-01' AND '2023-12-31';
8
```

Data Output Messages Notifications

CREATE VIEW










Query returned successfully in 39 msec.

SELECT \* FROM revenue;

Query Query History

```
1 SELECT * FROM revenue;
2
```

Data Output Messages Notifications

							
	revenue_amount 		numeric				
1	20114.35						



Запросы на модификацию данных:

- INSERT INTO

Добавить в таблицу “product” новый товар “mandarin”, где его цена варьируется от 150 до 200 рублей за килограмм, а первичный ключ генерируется через последовательность.

Фото таблицы до добавления элементов:

Query		Query History				
1		select * from product;				
Data Output		Messages				
	product_id [PK] character varying (10)	product_name character varying (100)	customer_id character varying (10)	ed_izm ed_izm	cost_ed integer	
110	product110	feijoa	customer3	килограмм	360	
111	product112	feijoa	customer4	килограмм	360	
112	product113	feijoa	customer5	килограмм	360	
113	product114	feijoa	customer6	килограмм	360	
114	product115	feijoa	customer7	килограмм	360	
115	product116	feijoa	customer8	килограмм	360	
116	product117	feijoa	customer9	килограмм	360	
117	product118	feijoa	customer10	килограмм	360	
118	product119	feijoa	customer11	килограмм	360	
119	product120	feijoa	customer12	килограмм	360	
120	product121	feijoa	customer13	килограмм	360	
121	product122	feijoa	customer14	килограмм	360	
122	product123	feijoa	customer15	килограмм	360	
123	product124	feijoa	customer16	килограмм	360	
124	product125	feijoa	customer17	килограмм	360	
125	product126	feijoa	customer18	килограмм	360	
126	product127	feijoa	customer19	килограмм	360	
127	product128	feijoa	customer20	килограмм	360	

Ход добавления:

1) Создаем последовательность

```
CREATE SEQUENCE product_seq START 140;
```

Query		Query History				
1		CREATE SEQUENCE product_seq START 140;				
2						
Data Output		Messages				
		ERROR: relation "product_seq" already exists				
		SQL state: 42P07				

2) Добавляем товар в таблицу:

```
INSERT INTO product (product_id, product_name, customer_id, ed_izm, cost_ed)
SELECT
```

```
    'product' || nextval('product_seq'),
    'mandarin', customer.customer_id,
    'килограмм', (random() * (200 - 150) + 150)::numeric(10,2)
```

```
FROM generate_series(1, 10) AS id
```

CROSS JOIN customer

WHERE customer.customer\_id IN (SELECT customer\_id FROM customer)

LIMIT 20;

Фото после добавления продукта:

Query		Query History			
1		select * from product;			
Data Output		Messages			
	product_id [PK] character varying (10)	product_name character varying (100)	customer_id character varying (10)	ed_izm ed_izm	cost_ed integer
117	product110	feijoa	customer2	килограмм	360
118	product111	feijoa	customer3	килограмм	360
119	product346	mandarin	customer11	килограмм	162
120	product347	mandarin	customer1	килограмм	193
121	product348	mandarin	customer5	килограмм	195
122	product349	mandarin	customer8	килограмм	170
123	product350	mandarin	customer20	килограмм	162
124	product351	mandarin	customer15	килограмм	191
125	product352	mandarin	customer16	килограмм	165
126	product353	mandarin	customer2	килограмм	165
127	product354	mandarin	customer4	килограмм	171
128	product112	feijoa	customer4	килограмм	360
129	product113	feijoa	customer5	килограмм	360
130	product114	feijoa	customer6	килограмм	360
131	product115	feijoa	customer7	килограмм	360
132	product116	feijoa	customer8	килограмм	360
133	product117	feijoa	customer9	килограмм	360
134	product118	feijoa	customer10	килограмм	360
135	product119	feijoa	customer11	килограмм	360
136	product120	feijoa	customer12	килограмм	360

- UPDATE

Обновить цены на товары “orange” и “feijoa” на 10% (скидка), где id в “customer\_id” нечетное число

Фото до обновления:

Query		Query History			
1		<b>select</b> * <b>from</b> product;			
2					
Data Output		Messages			
	product_id [PK] character varying (10)	product_name character varying (100)	customer_id character varying (10)	ed_izm ed_izm	cost_ed integer
126	product353	mandarin	customer2	килограмм	165
127	product354	mandarin	customer4	килограмм	171
128	product112	feijoa	customer4	килограмм	360
129	product113	feijoa	customer5	килограмм	360
130	product114	feijoa	customer6	килограмм	360
131	product115	feijoa	customer7	килограмм	360
132	product116	feijoa	customer8	килограмм	360
133	product117	feijoa	customer9	килограмм	360
134	product118	feijoa	customer10	килограмм	360
135	product119	feijoa	customer11	килограмм	360
136	product120	feijoa	customer12	килограмм	360
137	product121	feijoa	customer13	килограмм	360
138	product122	feijoa	customer14	килограмм	360
139	product123	feijoa	customer15	килограмм	360
140	product124	feijoa	customer16	килограмм	360
141	product125	feijoa	customer17	килограмм	360
142	product126	feijoa	customer18	килограмм	360
143	product127	feijoa	customer19	килограмм	360
144	product128	feijoa	customer20	килограмм	360
145	product355	mandarin	customer6	килограмм	196

Скрин запроса:

```
UPDATE product
SET cost_ed = cost_ed * 0.9
WHERE product_name IN ('orange', 'feijoa')
AND customer_id IN (
  SELECT customer_id
  FROM customer
  WHERE MOD(CAST(SUBSTRING(customer_name FROM '(\d+)') AS INTEGER), 2) = 1
);
```

Query		Query History			
1		<b>UPDATE</b> product			
2		<b>SET</b> cost_ed = cost_ed * 0.9			
3		<b>WHERE</b> product_name <b>IN</b> ('orange', 'feijoa')			
4		<b>AND</b> customer_id <b>IN</b> (			
5		<b>SELECT</b> customer_id			
6		<b>FROM</b> customer			
7		<b>WHERE</b> MOD(CAST(SUBSTRING(customer_name FROM '(\d+)') AS INTEGER), 2) = 1			
8		);			
9					
Data Output		Messages			
UPDATE 18					
Query returned successfully in 67 msec.					

Скрин после обновления:

Data Output Messages Notifications					
<div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div>					
	product_id [PK] character varying (10)	product_name character varying (100)	customer_id character varying (10)	ed_izm ed_izm	cost_ed integer
123	product343	mandarin	customer14	килограмм	176
124	product344	mandarin	customer18	килограмм	185
125	product345	mandarin	customer19	килограмм	198
126	product140	mandarin	customer8	килограмм	172
127	product56	orange	customer9	килограмм	230
128	product58	orange	customer9	килограмм	143
129	product111	feijoa	customer3	килограмм	324
130	product113	feijoa	customer5	килограмм	324
131	product115	feijoa	customer7	килограмм	324
132	product117	feijoa	customer9	килограмм	324
133	product119	feijoa	customer11	килограмм	324
134	product121	feijoa	customer13	килограмм	324
135	product123	feijoa	customer15	килограмм	324
136	product125	feijoa	customer17	килограмм	324
137	product127	feijoa	customer19	килограмм	324

• DELETE

Удалить поставщиков, у которых нет товаров или товары которых не попали в закупку

Фото до удаления:

Data Output Messages Notifications			
<div><div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div></div></div>			
	customer_id [PK] character varying (10)	customer_name character varying (100)	customer_address character varying (50)
7	customer13	caterer13	any street no.13
8	customer14	caterer14	any street no.14
9	customer15	caterer15	any street no.15
10	customer16	caterer16	any street no.16
11	customer17	caterer17	any street no.17
12	customer18	caterer18	any street no.18
13	customer19	caterer19	any street no.19
14	customer20	caterer20	any street no.20
15	customer21	catere21	any street no.21
16	customer22	catere22	any street no.22
17	customer23	catere23	any street no.23
18	customer24	catere24	any street no.24
19	customer25	catere25	any street no.25
20	customer26	catere26	any street no.26
21	customer27	catere27	any street no.27
22	customer28	catere28	any street no.28
23	customer29	catere29	any street no.29
24	customer30	catere30	any street no.30
25	customer6	caterer6	any street no.6
26	customer7	caterer7	any street no.7
27	customer1	new caterer	456 Market St
28	customer8	caterer8	any street no.8
29	customer9	caterer9	anv street no.9
Total rows: 30 of 30		Query complete 00:00:00.064	
			Ln

Запрос на удаление:




```

DELETE FROM customer
WHERE NOT EXISTS (
    SELECT 1
    FROM product
    WHERE customer.customer_id = product.customer_id
) AND NOT EXISTS (
    SELECT 1
    FROM purchase
    WHERE customer.customer_id = purchase.customer_id
);

```

Query	Query History
<pre> 1  DELETE FROM customer 2  WHERE NOT EXISTS ( 3      SELECT 1 4      FROM product 5      WHERE customer.customer_id = product.customer_id 6  ) AND NOT EXISTS ( 7      SELECT 1 8      FROM purchase 9      WHERE customer.customer_id = purchase.customer_id 10 ) ; 11 </pre>	
Data Output	Messages
DELETE 10	Query returned successfully in 57 msec.

Фото после удаления:

	<b>customer_id</b> [PK] character varying (10) 	<b>customer_name</b> character varying (100) 	<b>customer_address</b> character varying (50) 
1	customer11	caterer11	any street no.11
2	customer2	caterer2	any street no.2
3	customer3	caterer3	any street no.3
4	customer4	caterer4	any street no.4
5	customer5	caterer5	any street no.5
6	customer12	caterer12	any street no.12
7	customer13	caterer13	any street no.13
8	customer14	caterer14	any street no.14
9	customer15	caterer15	any street no.15
10	customer16	caterer16	any street no.16
11	customer17	caterer17	any street no.17
12	customer18	caterer18	any street no.18
13	customer19	caterer19	any street no.19
14	customer20	caterer20	any street no.20
15	customer6	caterer6	any street no.6
16	customer7	caterer7	any street no.7
17	customer1	new caterer	456 Market St
18	customer8	caterer8	any street no.8
19	customer9	caterer9	any street no.9
20	customer10	caterer10	any street no.10



Query Query History

1 **CREATE INDEX** product\_name\_idx **ON** product (product\_name);

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 63 msec.

```
SELECT customer_id, COUNT(DISTINCT product_name) AS num_products
FROM product
GROUP BY customer_id
ORDER BY num_products DESC
LIMIT 5;
```

Query Query History

1 **SELECT** customer\_id, **COUNT**(**DISTINCT** product\_name) **AS** num\_products  
 2 **FROM** product  
 3 **GROUP BY** customer\_id  
 4 **ORDER BY** num\_products **DESC**  
 5 **LIMIT** 5;

Data Output Messages Explain x Notifications

	customer_id character varying (10)	num_products bigint
1	customer5	12
2	customer1	12
3	customer10	12
4	customer3	12
5	customer8	10

Запрос выполнен за 44 миллисекунд. (с введением индексов)

Data Output Messages Explain x Notifications

Graphical Analysis Statistics

```

graph LR
    product[product] --> Sort1[Sort]
    Sort1 --> Aggregate[Aggregate]
    Aggregate --> Sort2[Sort]
    Sort2 --> Limit[Limit]
  
```

- 2) Вывести топ-3 товаров, которые были больше всего куплены за 2023 год
- ```
SELECT product.product_id, product.product_name
FROM product
JOIN purchase_composition ON product.product_id = purchase_composition.product_id
JOIN purchase ON purchase_composition.purchase_id = purchase.purchase_id
WHERE EXTRACT(YEAR FROM purchase.purchase_date) = 2023
GROUP BY product.product_id, product.product_name
ORDER BY COUNT(*) DESC
LIMIT 3;
```



Query Query History

```

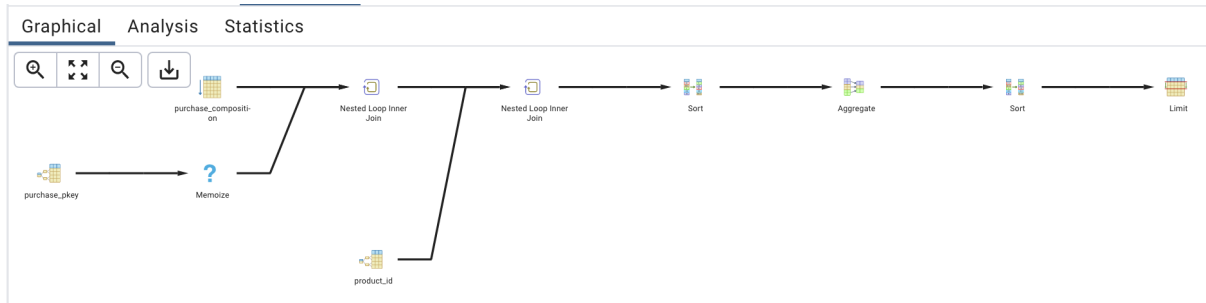
1 SELECT product.product_id, product.product_name
2 FROM product
3 JOIN purchase_composition ON product.product_id = purchase_composition.product_id
4 JOIN purchase ON purchase_composition.purchase_id = purchase.purchase_id
5 WHERE EXTRACT(YEAR FROM purchase.purchase_date) = 2023
6 GROUP BY product.product_id, product.product_name
7 ORDER BY COUNT(*) DESC
8 LIMIT 3;
9

```

Data Output Messages Explain X Notifications

|   | product_id<br>[PK] character varying (10) | product_name<br>character varying (100) |
|---|-------------------------------------------|-----------------------------------------|
| 1 | product110                                | feijoa                                  |
| 2 | product17                                 | water melon                             |
| 3 | product1                                  | pear                                    |

Запрос выполнен за 45 миллисекунд. (без введения индексов)



Создание индексов:

CREATE INDEX purchase\_date\_idx ON purchase (purchase\_date);

Query Query History

```

1 CREATE INDEX purchase_date_idx ON purchase (purchase_date);

```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 47 msec.

CREATE INDEX product\_id\_idx ON purchase\_composition (product\_id);

Query Query History

```

1 CREATE INDEX product_id_idx ON purchase_composition (product_id);

```

Data Output Messages Notifications

CREATE INDEX

Query returned successfully in 62 msec.

```

SELECT product.product_id, product.product_name
FROM product
JOIN purchase_composition ON product.product_id = purchase_composition.product_id
JOIN purchase ON purchase_composition.purchase_id = purchase.purchase_id
WHERE EXTRACT(YEAR FROM purchase.purchase_date) = 2023
GROUP BY product.product_id, product.product_name
ORDER BY COUNT(*) DESC
LIMIT 3;

```

wholesale\_base/postgres@PostgreSQL 14

Query Query History

```

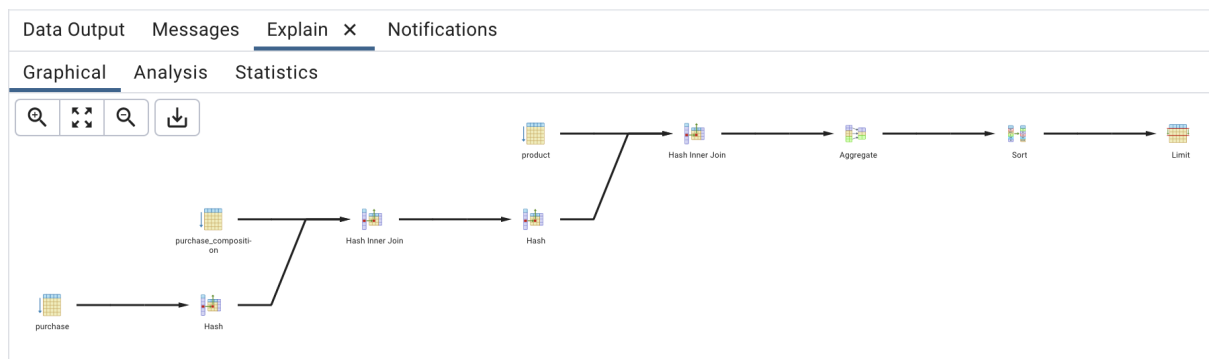
1 SELECT product.product_id, product.product_name
2 FROM product
3 JOIN purchase_composition ON product.product_id = purchase_composition.product_id
4 JOIN purchase ON purchase_composition.purchase_id = purchase.purchase_id
5 WHERE EXTRACT(YEAR FROM purchase.purchase_date) = 2023
6 GROUP BY product.product_id, product.product_name
7 ORDER BY COUNT(*) DESC
8 LIMIT 3;
9

```

Data Output Messages Explain x Notifications

|   | product_id<br>[PK] character varying (10) | product_name<br>character varying (100) |
|---|-------------------------------------------|-----------------------------------------|
| 1 | product17                                 | water melon                             |
| 2 | product110                                | feijoa                                  |
| 3 | product26                                 | compote                                 |

Запрос выполнен за 49 миллисекунд. (с введением индексов)



- 3) Вывести сколько в объеме товаров было вывезено за все закупки со складов
- ```

SELECT product.product_name, SUM(purchase_composition.amount_of_products)
FROM product
JOIN purchase_composition ON product.product_id = purchase_composition.product_id
GROUP BY product.product_name;

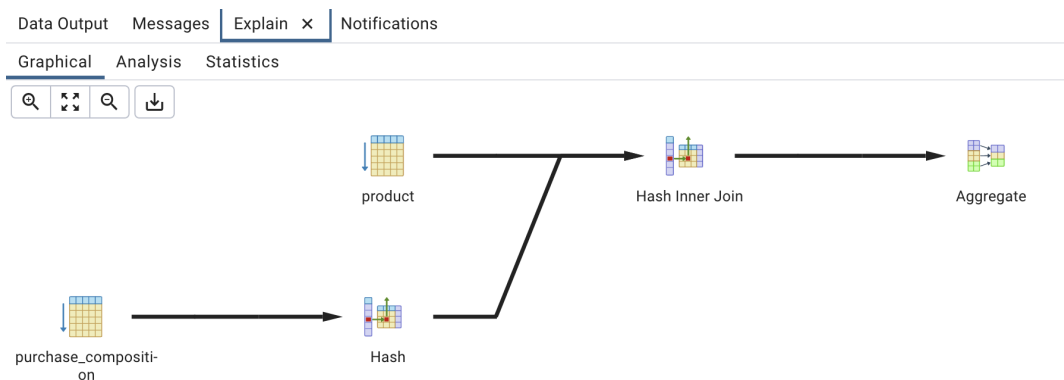
```

Query	Query History
1	<b>SELECT</b> product.product_name, <b>SUM</b> (purchase_composition.amount_of_products)
2	<b>FROM</b> product
3	<b>JOIN</b> purchase_composition <b>ON</b> product.product_id = purchase_composition.product_id
4	<b>GROUP BY</b> product.product_name;
5	

Data Output	Messages	Notifications
<div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>		
	product_name character varying (100)	sum bigint
1	water melon	92
2	dragon_fruit	136
3	melon	51
4	grape	44
5	feijoa	109
6	compote	12
7	orange	126
8	apple	139
9	juice	30
10	stawberry	102
11	pear	73

Запрос выполнен за 47 миллисекунд. (без введения индексов)



Создание индексов:

```
CREATE INDEX idx_product_id ON product(product_id);
```

Query	Query History
1	<b>CREATE INDEX</b> idx_product_id <b>ON</b> product(product_id);

Data Output	Messages	Notifications
CREATE INDEX		
Query returned successfully in 33 msec.		

CREATE INDEX idx\_product\_id ON purchase\_composition(product\_id);

Query Query History

---

1 **CREATE INDEX** idx\_product\_id **ON** purchase\_composition(product\_id);

---

Data Output Messages Notifications

---

ERROR: relation "idx\_product\_id" already exists  
SQL state: 42P07

SELECT product.product\_name, SUM(purchase\_composition.amount\_of\_products)  
FROM product  
JOIN purchase\_composition ON product.product\_id = purchase\_composition.product\_id  
GROUP BY product.product\_name;

Query Query History

---

1 **SELECT** product.product\_name, SUM(purchase\_composition.amount\_of\_products)  
2 **FROM** product  
3 **JOIN** purchase\_composition **ON** product.product\_id = purchase\_composition.product\_id  
4 **GROUP BY** product.product\_name;

---

Data Output Messages Explain X Notifications

---

Icons: [Menu] [Copy] [Paste] [Delete] [Refresh] [Download] [Zoom]

	product_name character varying (100)	sum bigint
1	water melon	92
2	dragon_fruit	136
3	melon	51
4	grape	44
5	feijoa	109
6	compote	12
7	orange	126
8	apple	139
9	juice	30
10	stawberry	102
11	pear	73

Запрос выполнен за 41 миллисекунд. (с введением индексов)

Data Output Messages Explain X Notifications

---

Graphical Analysis Statistics

---

Icons: [Zoom In] [Zoom Out] [Download]

```
graph LR; product --> HashInnerJoin[Hash Inner Join]; purchase_composition --> HashInnerJoin; HashInnerJoin --> Aggregate
```

The diagram illustrates the execution plan for the query. It starts with two input tables: 'product' and 'purchase\_composition'. Both tables are scanned and their data is loaded into a 'Hash' structure. These two hashed tables are then joined using a 'Hash Inner Join' operation. The result of the join is then passed to an 'Aggregate' operation to calculate the sum of 'amount\_of\_products' for each 'product\_name'.

Удаление индексов:

Query

Query History

1

SELECT indexname, tablename

2

FROM pg\_indexes

3

WHERE schemaname = 'public';

4

|

Data Output

Messages

Notifications

≡+

▼

	indexname name		tablename name	
8	product_id		product	
9	stock_pkey		stock	
10	passport		managers	
11	zapas_id		stock_in_stock	
12	orders_pkey		orders	
13	id		order_composition	
14	order_invoice_pkey		order_invoice	
15	purchase_pkey		purchase	
16	purchase_composition_pkey		purchase_composition	
17	purchase_invoice_pkey		purchase_invoice	
18	customer_id_idx		product	
19	product_name_idx		product	
20	purchase_date_idx		purchase	
21	product_id_idx		purchase_composition	
22	idx_product_id		product	

DROP INDEX product\_name\_idx;  
DROP INDEX customer\_id\_idx;  
DROP INDEX idx\_product\_id;  
DROP INDEX purchase\_date\_idx;  
DROP INDEX product\_id\_idx;

Query	Query History
1	<b>DROP INDEX</b> product_name_idx;
2	<b>DROP INDEX</b> customer_id_idx;
3	<b>DROP INDEX</b> idx_product_id;
4	<b>DROP INDEX</b> purchase_date_idx;
5	<b>DROP INDEX</b> product_id_idx;
6	

Data Output	Messages	Notifications
DROP INDEX		

```
SELECT indexname, tablename
FROM pg_indexes
WHERE schemaname = 'public';
```

Query	Query History
1	<b>SELECT</b> indexname, tablename
2	<b>FROM</b> pg_indexes
3	<b>WHERE</b> schemaname = 'public';
4	

Data Output	Messages	Notifications
<div> <div>≡</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🗑️</div> <div>📥</div> <div>📥</div> </div>		
indexname name	tablename name	
1	buyer_pkey	buyer
2	product_pkey	product
3	customer_pkey	customer
4	managers_pkey	managers
5	buyer_id	buyer
6	customer_id	customer
7	manager_id	managers
8	product_id	product
9	stock_pkey	stock
10	passport	managers
11	zapas_id	stock_in_stock
12	orders_pkey	orders
13	id	order_composition
14	order_invoice_pkey	order_invoice
15	purchase_pkey	purchase
16	purchase_composition_pkey	purchase_composition
17	purchase_invoice_pkey	purchase_invoice

**Вывод:**

Работа с запросами и модификацией данных является неотъемлемой частью работы с базами данных. Эффективность работы с базами данных напрямую зависит от умения

правильно использовать запросы и модифицировать данные. Индексы, в свою очередь, позволяют ускорить выполнение запросов в базе данных, что особенно важно для больших объемов данных. При создании индексов следует учитывать особенности запросов, которые будут выполняться на базе данных, и выбирать подходящий тип индекса. Однако, следует помнить, что слишком большое количество индексов может негативно сказаться на производительности базы данных, поэтому необходимо находить баланс между количеством индексов и скоростью выполнения запросов.