



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

**Институт информационных технологий (ИИТ)
Кафедра цифровой трансформации (ЦТ)**

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ
по дисциплине «Разработка баз данных»

Практическое занятие №1

Студенты группы *ИКБО-20-23 Комисарик М.А.*
.

(подпись)

Ассистент *Брайловский А.В.*

(подпись)

Отчет представлен «__» _____ 2025 г.

Москва 2025 г.

ПОСТАНОВКА ЗАДАЧИ

Цель работы: формирование и закрепление у студентов фундаментальных навыков работы с реляционными базами данных на примере СУБД Postgres Pro.

Постановка задачи: выполнить следующие шаги, основываясь на логической модели данных, которая была спроектирована в рамках курса «Проектирование баз данных» в предыдущем семестре:

1. На основе логической модели данных, созданной в прошлом семестре, письменно описать не менее 5 различных бизнес-правил и не менее 3 ограничений целостности для таблиц. Выбор бизнес-правил и ограничений целостности производится на усмотрение студента. Результаты представить в виде таблицы.
2. С использованием DDL-оператора CREATE TABLE создать все необходимые таблицы (согласно созданной в прошлом семестре логической модели данных) в СУБД Postgres Pro, корректно реализовав все описанные ограничения целостности.
3. Заполнить созданные таблицы согласованными тестовыми данными (не менее 5-7 записей на таблицу, где это применимо) с помощью оператора INSERT INTO.
4. Составить и выполнить не менее 6 SQL-запросов к таблицам, иллюстрирующих использование различных операторов SELECT и WHERE, согласно перечню, указанному в задании (см. Ход выполнения работы). В составленных запросах должны быть использованы все приведённые примеры.
5. Составить и выполнить по два SQL-запроса к таблицам для демонстрации работы предложений ORDER BY, GROUP BY и HAVING.
6. Каждый SQL-запрос сопровождать комментарием, объясняющим его назначение и логику работы.

ХОД РАБОТЫ

1 Анализ и описание ограничений целостности

В таблице 1 расположены 5 столбцов из таблицы "_order", для каждого из которых указано ограничение целостности и бизнес-правило.

Таблица 1 – Ограничения целостности таблицы _order

Название столбца	Тип данных	Ограничение	Обоснование (Бизнес-правило)
id_order	SERIAL	PRIMARY KEY	Уникальный идентификатор заказа, генерируется автоматически.
id_client	INT	FOREIGN KEY (client)	Ссылка на пользователя. Заказ не может существовать без пользователя.
id_delivery_address	INT	FOREIGN KEY (address)	Ссылка на адрес доставки. Заказ не может существовать без адреса доставки
order_amount	MONEY	NOT NULL	Стоимость заказа. Любой заказ имеет стоимость, составляемую из цен входящих в него блюд.
date_of_formation	DATE	NOT NULL	Дата формирования заказа. Любой заказ имеет дату своего формирования.

2 Создание структуры данных

На рисунке 1 представлен фрагмент кода запроса для создания всех таблиц и добавления всех ограничений целостности.

```

document_link varchar(50) NOT NULL
);

--
-- CREATE TABLE storage (
--   id_storage serial NOT NULL PRIMARY KEY,
--   phone_number varchar(50) NOT NULL,
--   id_address integer NOT NULL
-- );
--
-- CREATE TABLE ingredient_supplier (
--   id_ingredient_supplier serial NOT NULL PRIMARY KEY,
--   id_ingredient integer NOT NULL,
--   id_supplier integer NOT NULL
-- );
--
-- CREATE TABLE delivery_method (
--   id_delivery_method smallserial NOT NULL PRIMARY KEY,
--   name varchar(50) NOT NULL
-- );
--
ALTER TABLE app_rating ADD CONSTRAINT app_rating_id_client_fk FOREIGN KEY (id_client) REFERENCES client (id_client);
ALTER TABLE client ADD CONSTRAINT client_id_delivery_address_fk FOREIGN KEY (id_delivery_address) REFERENCES address (id_address);
ALTER TABLE employee_document ADD CONSTRAINT employee_document_id_document_fk FOREIGN KEY (id_document) REFERENCES document (id_document);
ALTER TABLE employee_document ADD CONSTRAINT employee_document_id_employee_fk FOREIGN KEY (id_employee) REFERENCES employee (id_employee);
ALTER TABLE employee ADD CONSTRAINT employee_id_job_position_fk FOREIGN KEY (id_job_position) REFERENCES job_position (id_job_position);
ALTER TABLE employee ADD CONSTRAINT employee_id_registration_address_fk FOREIGN KEY (id_registration_address) REFERENCES address (id_address);
ALTER TABLE ingredient_supplier ADD CONSTRAINT ingredient_supplier_id_ingredient_fk FOREIGN KEY (id_ingredient) REFERENCES ingredient (id_ingredient);
ALTER TABLE ingredient_supplier ADD CONSTRAINT ingredient_supplier_id_supplier_fk FOREIGN KEY (id_supplier) REFERENCES supplier (id_supplier);
ALTER TABLE _order ADD CONSTRAINT order_delivery_method_id_fk FOREIGN KEY (delivery_method_id) REFERENCES delivery_method (id_delivery_method);
ALTER TABLE order_document ADD CONSTRAINT order_document_id_document_fk FOREIGN KEY (id_document) REFERENCES document (id_document);
ALTER TABLE order_document ADD CONSTRAINT order_document_id_order_fk FOREIGN KEY (id_order) REFERENCES _order (id_order);

```

Name	Value
Queries	58
Updated Rows	0
Execute time	1.598s
Fetch time	0.000s
Total time	1.598s
Start time	2025-09-07 21:25:53.260
Finish time	2025-09-07 21:25:54.919

Рисунок 1 – Создание таблиц и добавление зависимостей

3 Заполнение таблиц данными

На рисунке 2 представлен фрагмент кода запроса для заполнения таблиц данными.

```

(1, 'Alice', 'Johnson', NULL, 'emp_hash_1', '555-3333', 1, '2023-01-15', '2025-01-15'),
(2, 'Bob', 'Williams', 'Lee', 'emp_hash_2', '555-4444', 2, '2023-02-20', '2025-02-20'),
(3, 'Carol', 'Martinez', NULL, 'emp_hash_3', '555-8888', 3, '2023-03-10', '2024-03-10'),
(4, 'Dave', 'Anderson', 'Paul', 'emp_hash_4', '555-9999', 4, '2023-04-05', '2024-10-05'),
(5, 'Eva', 'Garcia', 'Maria', 'emp_hash_5', '555-0000', 5, '2023-05-12', '2025-05-12'),
(2, 'Frank', 'Taylor', NULL, 'emp_hash_6', '555-1212', 6, '2023-06-18', '2024-12-18'),
(3, 'Grace', 'Thomas', 'Elizabeth', 'emp_hash_7', '555-1313', 7, '2023-07-22', '2024-07-22');

--
-- INSERT INTO _order (id_client, id_delivery_address, order_amount, date_of_formation, delivery_method_id, status_id) VALUES
-- (1, 1, '25.97', '2023-10-05', 2, 3),
-- (2, 2, '15.98', '2023-10-06', 1, 2),
-- (3, 3, '32.97', '2023-10-07', 2, 3),
-- (4, 4, '18.98', '2023-10-08', 1, 1),
-- (5, 5, '45.96', '2023-10-09', 2, 2),
-- (6, 6, '12.99', '2023-10-10', 1, 3),
-- (7, 7, '28.97', '2023-10-11', 1, 4);
--
-- INSERT INTO product_category (id_product, id_category) VALUES
-- (1, 1),
-- (2, 1),
-- (3, 2),
-- (4, 1),
-- (5, 4),
-- (6, 5),
-- (7, 3);
--
-- INSERT INTO product_ingredient (id_product, id_ingredient, ingredient_weight) VALUES
-- (1, 1, 500),
-- (1, 2, 200),
-- (1, 3, 150),
-- (2, 1, 500),
-- (2, 2, 200),
-- (2, 3, 150),
-- (2, 4, 100),
-- ...

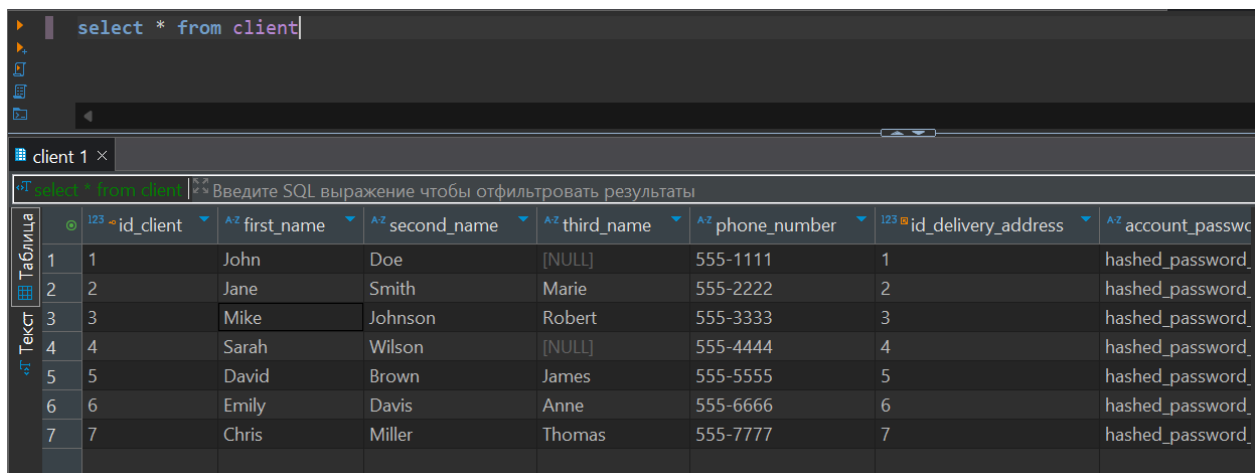
```

Name	Value
Queries	26
Updated Rows	169
Execute time	0.921s
Fetch time	0.000s
Total time	0.921s
Start time	2025-09-07 22:19:09.940
Finish time	2025-09-07 22:19:10.907

Рисунок 2 – Заполнение таблиц данными

4 Составление запросов на выборку (часть 1)

На рисунке 3 представлен запрос на получение всех строк таблицы client.

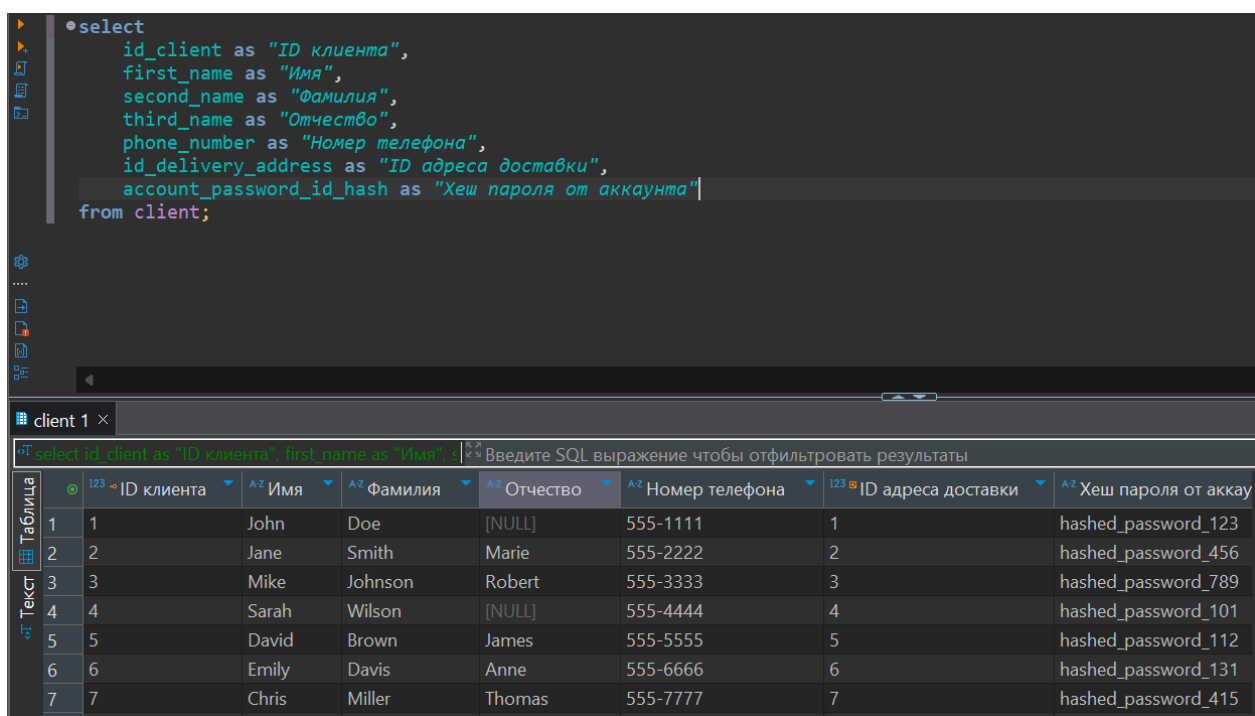


The screenshot shows a SQL client window with a query editor at the top containing the text `select * from client`. Below the editor, a tab labeled 'client 1' is active. The results are displayed in a table view with columns: `id_client`, `first_name`, `second_name`, `third_name`, `phone_number`, `id_delivery_address`, and `account_password`. The table contains 7 rows of data.

	<code>id_client</code>	<code>first_name</code>	<code>second_name</code>	<code>third_name</code>	<code>phone_number</code>	<code>id_delivery_address</code>	<code>account_password</code>
1	1	John	Doe	[NULL]	555-1111	1	hashed_password_123
2	2	Jane	Smith	Marie	555-2222	2	hashed_password_456
3	3	Mike	Johnson	Robert	555-3333	3	hashed_password_789
4	4	Sarah	Wilson	[NULL]	555-4444	4	hashed_password_101
5	5	David	Brown	James	555-5555	5	hashed_password_112
6	6	Emily	Davis	Anne	555-6666	6	hashed_password_131
7	7	Chris	Miller	Thomas	555-7777	7	hashed_password_415

Рисунок 3 – Первый запрос из пункта 4

На рисунке 4 представлен запрос на получение всех строк таблицы client с переименованием всех имен столбцов на выходе.



The screenshot shows a SQL client window with a query editor at the top containing the text `select id_client as 'ID клиента', first_name as 'Имя', second_name as 'Фамилия', third_name as 'Отчество', phone_number as 'Номер телефона', id_delivery_address as 'ID адреса доставки', account_password_id_hash as 'Хеш пароля от аккаунта' from client;`. Below the editor, a tab labeled 'client 1' is active. The results are displayed in a table view with columns: `ID клиента`, `Имя`, `Фамилия`, `Отчество`, `Номер телефона`, `ID адреса доставки`, and `Хеш пароля от аккаунта`. The table contains 7 rows of data.

	<code>ID клиента</code>	<code>Имя</code>	<code>Фамилия</code>	<code>Отчество</code>	<code>Номер телефона</code>	<code>ID адреса доставки</code>	<code>Хеш пароля от аккаунта</code>
1	1	John	Doe	[NULL]	555-1111	1	hashed_password_123
2	2	Jane	Smith	Marie	555-2222	2	hashed_password_456
3	3	Mike	Johnson	Robert	555-3333	3	hashed_password_789
4	4	Sarah	Wilson	[NULL]	555-4444	4	hashed_password_101
5	5	David	Brown	James	555-5555	5	hashed_password_112
6	6	Emily	Davis	Anne	555-6666	6	hashed_password_131
7	7	Chris	Miller	Thomas	555-7777	7	hashed_password_415

Рисунок 4 – Второй запрос из пункта 4

На рисунке 5 представлен запрос на получение всех строк таблицы client, в колонке "first_name" которых находится строка 'John'.

The screenshot shows a database client interface. At the top, a SQL query is entered: `select * from client where first_name = 'John'`. Below the query editor, a tab labeled "client 1" is active. The results are displayed in a table with the following columns: `id_client`, `first_name`, `second_name`, `third_name`, `phone_number`, `id_delivery_address`, and `account_password_id_hash`. The results table contains one row with the following values: 1, John, Doe, [NULL], 555-1111, 1, and hashed_password_123.

	id_client	first_name	second_name	third_name	phone_number	id_delivery_address	account_password_id_hash
1	1	John	Doe	[NULL]	555-1111	1	hashed_password_123

Рисунок 5 – Третий запрос из пункта 4

На рисунке 6 представлен запрос на получение всех строк таблицы product, в колонке "price" которых находится денежная сумма в пределах от 5 до 12 долларов.

The screenshot shows a database client interface. At the top, a SQL query is entered: `select * from product where price::numeric between 5.0 and 12.0;`. Below the query editor, a tab labeled "product 1" is active. The results are displayed in a table with the following columns: `id_product`, `name`, `description`, and `price`. The results table contains three rows with the following values: (1, Margherita Pizza, Classic pizza with tomato and cheese, \$10.99), (5, Chocolate Cake, Rich chocolate dessert, \$6.99), and (6, Caesar Salad, Fresh salad with Caesar dressing, \$8.99).

	id_product	name	description	price
1	1	Margherita Pizza	Classic pizza with tomato and cheese	\$10.99
2	5	Chocolate Cake	Rich chocolate dessert	\$6.99
3	6	Caesar Salad	Fresh salad with Caesar dressing	\$8.99

Рисунок 6 – Четвертый запрос из пункта 4

На рисунке 7 представлен запрос на получение всех строк таблицы client, в колонке "first_name" которых находится одна из строк 'John', 'Myla', 'Mike', 'Eve', 'Emily'.

The screenshot shows a database client interface. At the top, a SQL query is entered: `select * from client where first_name in ('John', 'Myla', 'Mike', 'Eve', 'Emily');`. Below the query editor, a tab labeled "client 1" is active. The results are displayed in a table with the following columns: `id_client`, `first_name`, `second_name`, `third_name`, `phone_number`, `id_delivery_address`, and `account_password_id_hash`. The results table contains three rows with the following values: (1, John, Doe, [NULL], 555-1111, 1, hashed_password_123), (3, Mike, Johnson, Robert, 555-3333, 3, hashed_password_789), and (6, Emily, Davis, Anne, 555-6666, 6, hashed_password_131).

	id_client	first_name	second_name	third_name	phone_number	id_delivery_address	account_password_id_hash
1	1	John	Doe	[NULL]	555-1111	1	hashed_password_123
2	3	Mike	Johnson	Robert	555-3333	3	hashed_password_789
3	6	Emily	Davis	Anne	555-6666	6	hashed_password_131

Рисунок 7 – Пятый запрос из пункта 4

На рисунке 8 представлен запрос на получение всех строк таблицы client, в колонке "third_name" которых находится нулевой элемент.

```
select *
from client
where third_name is null;
```

	id_client	first_name	second_name	third_name	phone_number	id_delivery_address	account_password_id_hash
1	1	John	Doe	[NULL]	555-1111	1	hashed_password_123
2	4	Sarah	Wilson	[NULL]	555-4444	4	hashed_password_101

Рисунок 8 – Шестой запрос из пункта 4

5 Составление запросов на выборку (часть 2)

На рисунке 9 представлен запрос на получение всех строк таблицы product, отсортированных по значению столбца "price" в возрастающем порядке.

```
select *
from product
order by
price
```

	id_product	name	description	price
1	3	Cola	Refreshing soft drink	\$2.99
2	7	Garlic Bread	Toasted bread with g.	\$4.99
3	5	Chocolate Ca	Rich chocolate dessert	\$6.99
4	6	Caesar Salad	Fresh salad with Caesi	\$8.99
5	1	Margherita P	Classic pizza with torr	\$10.99
6	2	Pepperoni Pi	Pizza with pepperoni	\$12.99
7	4	Veggie Supre	Pizza with assorted ve	\$14.99

Рисунок 9 – Первый запрос из пункта 5

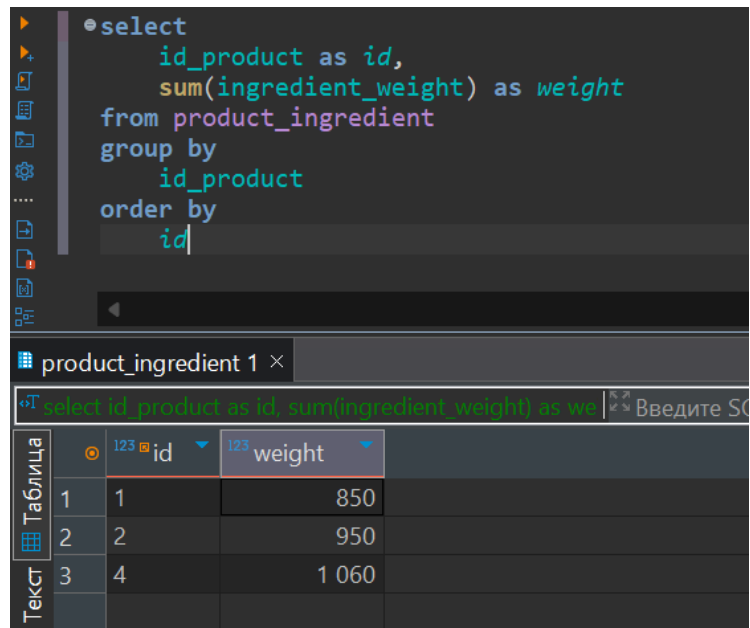
На рисунке 10 представлен запрос на получение всех строк таблицы product, отсортированных по значению столбца "price" в убывающем порядке.

```
select *
from product
order by
price desc
```

	id_product	name	description	price
1	4	Veggie Supreme	Pizza with assorted vegetables	\$14.99
2	2	Pepperoni Pizza	Pizza with pepperoni slices	\$12.99
3	1	Margherita Pizza	Classic pizza with tomato and c	\$10.99
4	6	Caesar Salad	Fresh salad with Caesar dressin	\$8.99
5	5	Chocolate Cake	Rich chocolate dessert	\$6.99
6	7	Garlic Bread	Toasted bread with garlic butte	\$4.99
7	3	Cola	Refreshing soft drink	\$2.99

Рисунок 10 – Второй запрос из пункта 5

На рисунке 11 представлен запрос на получение веса каждого продукта, составленного из весов каждого входящего ингредиента.

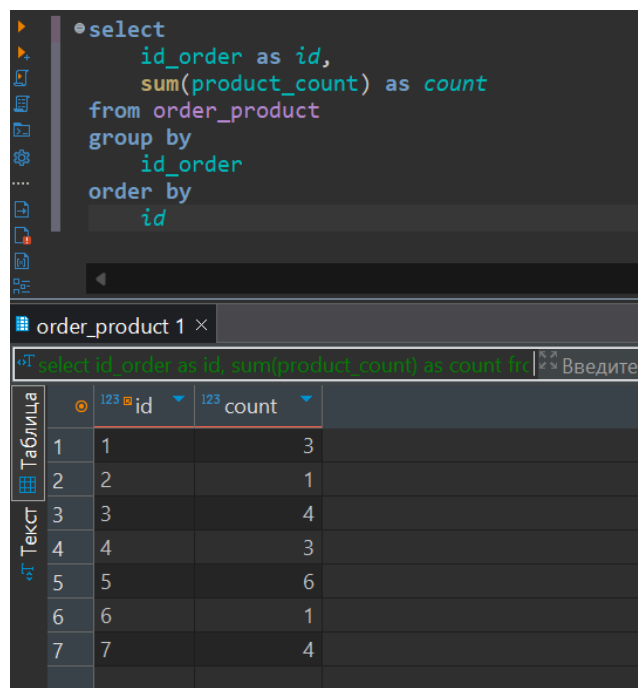


```
select
  id_product as id,
  sum(ingredient_weight) as weight
from product_ingredient
group by
  id_product
order by
  id
```

	id	weight
1	1	850
2	2	950
3	4	1 060

Рисунок 11 – Третий запрос из пункта 5

На рисунке 12 представлен запрос на получение количества входящих блюд (не уникальных) в каждый заказ.



```
select
  id_order as id,
  sum(product_count) as count
from order_product
group by
  id_order
order by
  id
```

	id	count
1	1	3
2	2	1
3	3	4
4	4	3
5	5	6
6	6	1
7	7	4

Рисунок 12 – Четвертый запрос из пункта 5

На рисунке 13 представлен запрос на получение всех заказов, которые включают в себя более 1 уникального блюда.

```
select
    id_order as id,
    count(*) as number_of_products
from order_product
group by
    id_order
having
    count(*) > 1
order by
```

order_product 1 ×

select id_order as id, count(*) as number_of_products | Введите SQL

	id	number_of_products
1	1	2
2	3	2
3	4	2
4	5	2
5	7	3

Рисунок 13 – Пятый запрос из пункта 5

На рисунке 14 представлен запрос на получение всех блюд, которые весят меньше килограмма.

```
select
    id_product as id,
    sum(ingredient_weight) as weight
from product_ingredient
group by
    id_product
having
    sum(ingredient_weight) < 1000
order by
    id
```

product_ingredient 1 ×

select id_product as id, sum(ingredient_weight) as weight | Введите SQL

	id	weight
1	1	850
2	2	950

Рисунок 14 – Шестой запрос из пункта 5