

Домашнее задание 2. Основные операторы PostgreSQL

Даны CSV-файлы:

customer.csv

product.csv

orders.csv

order_items.csv

Предварительный экспресс анализ данных выполнен с помощью pandas. Результаты сохранены в файле `pandas_analysis_hw2.ipynb`. В результате было выявлено, что из 190 `product_id` лишь 101 являются уникальными, поэтому `product_id` в таблице **product** не будет являться `primary key`. Кроме того, с помощью такого экспресс анализа удалось определиться с типами данных для создания соответствующих таблиц.

Создаем таблицы с перечисленными ниже структурами, используя CSV-файлы.

Таблица **customer**:

Поле	Описание	Тип данных
<code>customer_id</code>	ID клиента	<code>integer [primary key]</code>
<code>first_name</code>	Имя клиента	<code>varchar(50)</code>
<code>last_name</code>	Фамилия клиента	<code>varchar(50)</code>
<code>gender</code>	Пол	<code>varchar(10)</code>
<code>DOB</code>	Дата рождения	<code>date</code>
<code>job_title</code>	Профессия	<code>varchar(100)</code>
<code>job_industry_category</code>	Сфера деятельности	<code>varchar(50)</code>
<code>wealth_segment</code>	Сегмент благосостояния	<code>varchar(20)</code>
<code>deceased_indicator</code>	Индикатор актуального клиента	<code>varchar(1)</code>
<code>owns_car</code>	Индикатор наличия автомобиля	<code>varchar(3)</code>
<code>address</code>	Адрес проживания	<code>varchar(200)</code>
<code>postcode</code>	Почтовый индекс	<code>integer</code>
<code>state</code>	Штаты	<code>varchar(20)</code>
<code>country</code>	Страна проживания	<code>varchar(20)</code>
<code>property_valuation</code>	Оценка имущества	<code>integer</code>

Таблица **product**:

Поле	Описание	Тип данных
product_id	ID продукта	integer
brand	Бренд	varchar(50)
product_line	Линейка продуктов	varchar(20)
product_class	Класс продукта	varchar(10)
product_size	Размер продукта	varchar(10)
list_price	Цена	decimal(10,2)
standard_cost	Стандартная стоимость	decimal(10,2)

Таблица **orders**:

Поле	Описание	Тип данных
order_id	ID транзакции	integer [primary key]
customer_id	ID клиента	integer
order_date	Дата транзакции	date
online_order	Индикатор онлайн-заказа	boolean
order_status	Статус транзакции	varchar(10)

Таблица **order_items**:

Поле	Описание	Тип данных
order_item_id	ID позиции в заказе	integer [primary key]
order_id	ID заказа	integer
product_id	ID продукта	integer
quantity	Количество данного продукта в заказе	decimal(10,2)
item_list_price_at_sale	Цена продукта в момент продажи	decimal(10,2)
item_standard_cost_at_sale	Стандартная стоимость продукта в момент продажи	decimal(10,2)

Создание таблиц и просмотр загруженных из файлов данных приведены ниже на рисунках 1 – 5.

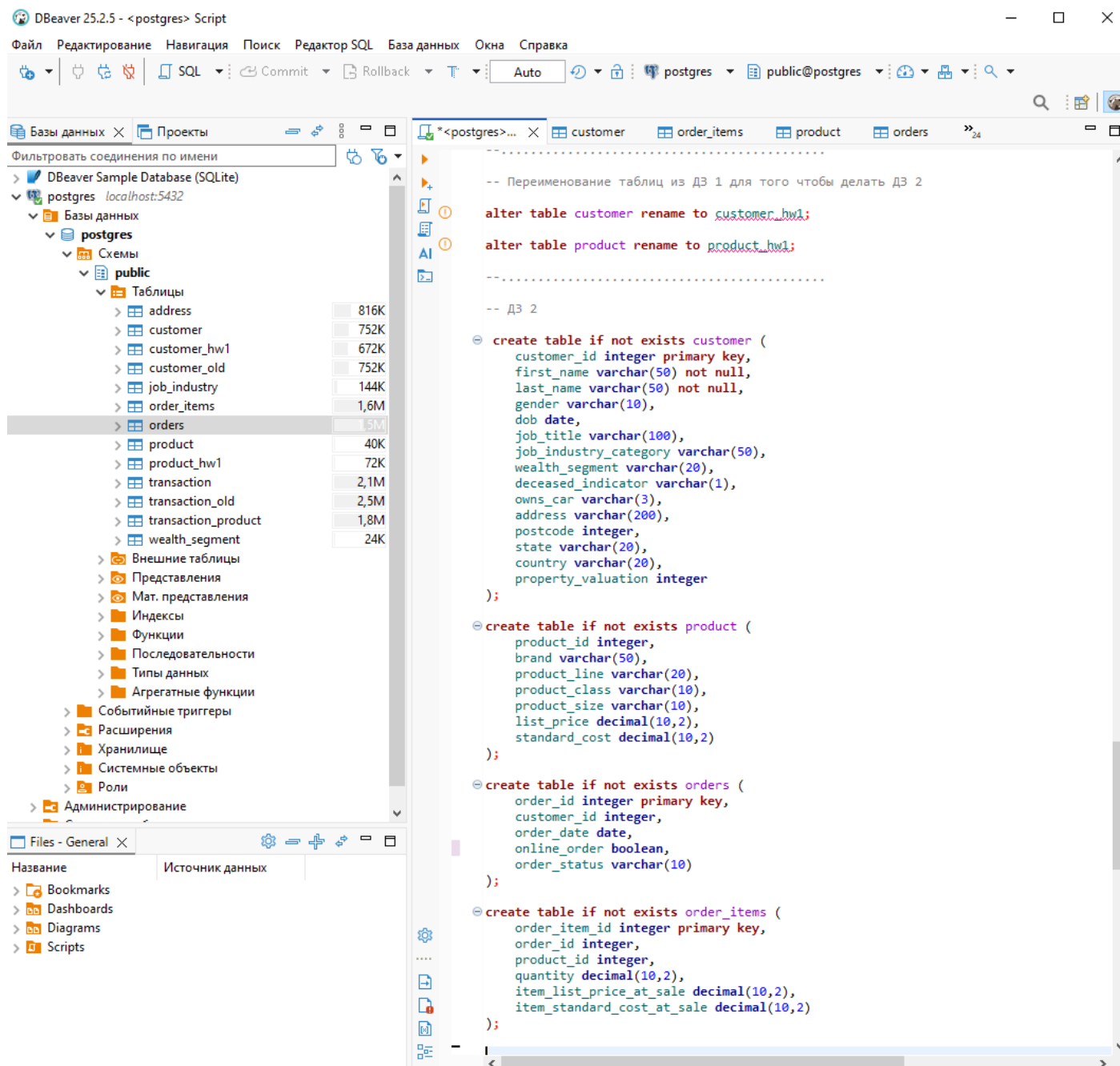


Рисунок 1 – Создание таблиц **customer**, **product**, **orders**, **order_items**

DBaver 25.2.5 - customer

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

SQL Commit Rollback Auto postgres public@postgres

Базы данных × Проекты

Фильтровать соединения по имени

DBaver Sample Database (SQLite)

postgres localhost:5432

Базы данных

postgres

Схемы

public

Таблицы

address 816K

customer 752K

Колонки

123 customer_id (int4)

AZ first_name (varchar(50))

AZ last_name (varchar(50))

AZ gender (varchar(10))

dob (date)

AZ job_title (varchar(100))

AZ job_industry_category (varchar(50))

AZ wealth_segment (varchar(20))

AZ deceased_indicator (varchar(1))

AZ owns_car (varchar(3))

AZ address (varchar(200))

123 postcode (int4)

AZ state (varchar(20))

AZ country (varchar(20))

123 property_valuation (int4)

Ограничения

Внешние ключи

Индексы

Зависимости

Свойства Данные Диаграмма

Показать SQL Введите SQL выражение чтобы отфильтровать результаты

Таблица	123 customer_id	AZ first_name	AZ last_name	AZ gender	dob
1	1	Laraine	Medendorp	F	1953-
2	2	Eli	Bockman	Male	1980-
3	3	Arlin	Dearle	Male	1954-
4	4	Talbot		Male	1961-
5	5	Sheila-kathryn	Calton	Female	1977-
6	6	Curr	Duckhouse	Male	1966-
7	7	Fina	Merali	Female	1976-
8	8	Rod	Inder	Male	1962-
9	9	Mala	Lind	Female	1973-
10	10	Fiorenze	Birdall	Female	1988-
11	11	Uriah	Bisatt	Male	1954-
12	12	Sawyer	Flattman	Male	1994-
13	13	Gabriele	Norcross	Male	1955-
14	14	Rayshell	Kitterman	Female	1983-
15	15	Erroll	Radage	Male	2000-
16	16	Harlin	Parr	Male	1977-
17	17	Heath	Faraday	Male	1962-
18	18	Marjie	Neasham	Female	1967-
19	19	Sorcha	Keyson	Female	2001-
20	20	Basile	Firth	Male	1980-
21	21	Mile	Cammocke	Male	1980-
22	22	Deeanne	Durtnell	Female	1962-
23	23	Olav	Polak	Male	1995-
24	24	Kim	Skpsey	Female	1977-

Рисунок 2 – Просмотр данных таблицы **customer**

DBaver 25.2.5 - product

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

SQL Commit Rollback Auto postgres public@postgres

Базы данных × Проекты

Фильтровать соединения по имени

DBaver Sample Database (SQLite)

postgres localhost:5432

Базы данных

postgres

Схемы

public

Таблицы

address 816K

customer 752K

customer_hw1 672K

customer_old 752K

job_industry 144K

order_items 1,6M

orders 1,5M

product 40K

Колонки

123 product_id (int4)

AZ brand (varchar(50))

AZ product_line (varchar(20))

AZ product_class (varchar(10))

AZ product_size (varchar(10))

123 list_price (numeric(10, 2))

123 standard_cost (numeric(10, 2))

Ограничения

Внешние ключи

Индексы

Зависимости

Ссылки

Секции таблиц

Свойства Данные Диаграмма

Показать SQL Введите SQL выражение чтобы отфильтровать результаты

Таблица	123 product_id	AZ brand	AZ product_line	AZ product_class	AZ prod
1	14	Trek Bicycles	Standard	medium	small
2	28	Norco Bicycles	Standard	medium	small
3	0	Solex	Standard	medium	medium
4	5	Giant Bicycles	Standard	high	medium
5	22	WeareA2B	Standard	medium	medium
6	0	OHM Cycles	Standard	high	medium
7	12	WeareA2B	Standard	medium	medium
8	53	OHM Cycles	Standard	medium	medium
9	7	Giant Bicycles	Standard	medium	small
10	6	OHM Cycles	Standard	high	medium
11	0	Giant Bicycles	Standard	medium	large
12	18	Norco Bicycles	Standard	high	medium
13	87	OHM Cycles	Standard	medium	medium
14	6	Solex	Standard	high	medium
15	84	Trek Bicycles	Road	medium	medium
16	0	WeareA2B	Standard	medium	small
17	48	WeareA2B	Standard	medium	medium
18	91	WeareA2B	Standard	low	medium
19	97	OHM Cycles	Road	medium	medium
20	82	Norco Bicycles	Standard	high	medium
21	87	Giant Bicycles	Standard	high	medium
22	28	Solex	Road	medium	small
23	62	Solex	Standard	medium	medium
24	60	Giant Bicycles	Standard	high	small

Рисунок 3 – Просмотр данных таблицы **product**

DBeaer 25.2.5 - orders

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

SQL Commit Rollback Auto postgres public@postgres

Базы данных Проекты

Фильтровать соединения по имени

DBeaer Sample Database (SQLite)

postgres localhost:5432

Базы данных

postgres

Схемы

public

Таблицы

address 816K

customer 752K

customer_hw1 672K

customer_old 752K

job_industry 144K

order_items 1,6M

orders 5M

Колонки

123 order_id (int4)

123 customer_id (int4)

order_date (date)

online_order (bool)

AZ order_status (varchar(10))

Ограничения

Внешние ключи

Индексы

Зависимости

Ссылки

Секции таблиц

Триггеры

Правила

Пolicies

Свойства Данные Диаграмма

Показать SQL Введите SQL выражение чтобы отфильтровать результаты

Таблица	123 order_id	123 customer_id	order_date	online_order	AZ on
1	1	2 950	2017-02-25	[]	Appro
2	2	3 120	2017-05-21	[v]	Appro
3	3	402	2017-10-16	[]	Appro
4	4	3 135	2017-08-31	[]	Appro
5	5	787	2017-10-01	[v]	Appro
6	6	2 339	2017-03-08	[v]	Appro
7	7	1 542	2017-04-21	[v]	Appro
8	8	2 459	2017-07-15	[]	Appro
9	9	1 305	2017-08-10	[]	Appro
10	10	3 262	2017-08-30	[v]	Appro
11	11	1 986	2017-01-17	[]	Appro
12	12	2 783	2017-01-05	[v]	Appro
13	13	1 243	2017-02-26	[v]	Appro
14	14	2 717	2017-09-10	[]	Appro
15	15	247	2017-06-11	[]	Appro
16	16	2 961	2017-10-10	[]	Appro
17	17	2 426	2017-04-03	[]	Appro
18	18	1 842	2017-06-02	[]	Appro
19	19	2 268	2017-04-06	[v]	Appro
20	20	3 002	2017-01-28	[v]	Appro
21	21	1 582	2017-10-09	[]	Appro
22	22	595	2017-06-29	[v]	Appro
23	23	2 001	2017-04-08	[v]	Appro
24	24	515	2017-10-18	[]	Appro

Рисунок 4 – Просмотр данных таблицы orders

DBeaer 25.2.5 - order_items

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

SQL Commit Rollback Auto postgres public@postgres

Базы данных Проекты

Фильтровать соединения по имени

DBeaer Sample Database (SQLite)

postgres localhost:5432

Базы данных

postgres

Схемы

public

Таблицы

address 816K

customer 752K

customer_hw1 672K

customer_old 752K

job_industry 144K

order_items 1,6M

Колонки

123 order_item_id (int4)

123 order_id (int4)

123 product_id (int4)

123 quantity (numeric(10, 2))

123 item_list_price_at_sale (numeric(10, 2))

123 item_standard_cost_at_sale (numeric(10, 2))

AZ order_item_id, "order_id", "product_id"

Ограничения

Внешние ключи

Индексы

Зависимости

Ссылки

Секции таблиц

Триггеры

Правила

Свойства Данные Диаграмма

Показать SQL Введите SQL выражение чтобы отфильтровать результаты

Таблица	123 order_item_id	123 order_id	123 product_id	123 quantity	123 item
1	1	1	2	6	6
2	2	2	3	2	2
3	3	3	37	7	7
4	4	4	88	4	4
5	5	5	78	7	7
6	6	6	25	5	5
7	7	7	22	3	3
8	8	8	15	10	10
9	9	9	67	7	7
10	10	10	12	2	2
11	11	11	5	8	8
12	12	12	61	1	1
13	13	13	35	8	8
14	14	14	16	3	3
15	15	15	12	7	7
16	16	16	3	4	4
17	17	17	79	10	10
18	18	18	33	6	6
19	19	19	54	6	6
20	20	20	25	2	2
21	21	21	27	3	3
22	22	22	37	9	9
23	23	23	37	9	9
24	24	24	82	1	1

Рисунок 5 – Просмотр данных таблицы order_items

Выполнение запросов

1. Вывести все уникальные бренды, у которых есть хотя бы один продукт со стандартной стоимостью выше 1500 долларов, и суммарными продажами не менее 1000 единиц.

The screenshot shows the DBeaver 25.2.5 interface with a PostgreSQL database. The left sidebar displays the database schema, including tables like 'orders' and 'product'. The main editor shows a SQL query that filters products based on their standard cost and total sales. The bottom panel displays the results of the query, showing two unique brands: 'Giant Bicycles' and 'OHM Cycles'.

```
-- 1. Вывести все уникальные бренды, у которых есть хотя бы один продукт
-- со стандартной стоимостью выше 1500 долларов,
-- и суммарными продажами не менее 1000 единиц.

select distinct
  product.brand
from
  product
where
  product.standard_cost > 1500
  and product.product_id in (
    select
      order_items.product_id
    from
      order_items
    group by
      order_items.product_id
    having
      sum(order_items.quantity) >= 1000
  );
```

product_id	brand
1	Giant Bicycles
2	OHM Cycles

2 строк получено - 0.0s, 2025-11-23 в 19:40:29

Рисунок 6 – Реализация запроса № 1

2. Для каждого дня в диапазоне с 2017-04-01 по 2017-04-09 включительно вывести количество подтвержденных онлайн-заказов и количество уникальных клиентов, совершивших эти заказы.

The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database structure for a PostgreSQL instance at localhost:5432, specifically the 'public' schema. The 'orders' table is highlighted, showing its columns: order_id (int4), customer_id (int4), order_date (date), online_order (bool), and order_status (varchar(10)).

The main editor window contains a SQL query for the 'orders' table:

```
-- 2. Для каждого дня в диапазоне с 2017-04-01 по 2017-04-09 включительно
-- вывести количество подтвержденных онлайн-заказов и количество уникальных
-- клиентов, совершивших эти заказы.

select
    order_date,
    count(*) as number_of_confirmed_online_orders,
    count(distinct customer_id) as number_of_unique_customers
from
    orders
where
    order_date between '2017-04-01' and '2017-04-09'
    and online_order = true
    and order_status = 'Approved'
group by
    order_date
order by
    order_date;
```

Below the query editor, the results are displayed in a table view for 'orders 1'. The table has three columns: 'order_date', 'number_of_confirmed_online_orders', and 'number_of_unique_customers'. The data shows the count of confirmed orders and unique customers for each day from April 1st to April 9th, 2017.

order_date	number_of_confirmed_online_orders	number_of_unique_customers
2017-04-01	37	37
2017-04-02	29	29
2017-04-03	27	27
2017-04-04	32	32
2017-04-05	33	32
2017-04-06	36	36
2017-04-07	24	24
2017-04-08	33	33
2017-04-09	30	30

The bottom status bar indicates that 9 rows were retrieved in 0.0s on 2025-11-23 at 14:36:42.

Рисунок 7 – Реализация запроса № 2

3. Вывести профессии клиентов:

- из сферы IT, чья профессия начинается с Senior;
- из сферы Financial Services, чья профессия начинается с Lead.

Для обеих групп учитывать только клиентов старше 35 лет. Объединить выборки с помощью UNION ALL.

The screenshot shows the DBeaver 25.2.5 interface. The main window displays a SQL script for querying client professions. The script is as follows:

```
-- 3. Вывести профессии клиентов:  
-- - из сферы IT, чья профессия начинается с Senior;  
-- - из сферы Financial Services, чья профессия начинается с Lead.  
-- Для обеих групп учитывать только клиентов старше 35 лет.  
-- Объединить выборки с помощью UNION ALL.  
  
select distinct  
    job_title as profession  
from  
    customer  
where  
    job_industry_category = 'IT'  
    and job_title like 'Senior%'  
    and extract(year from age(current_date, dob)) > 35  
  
union all  
  
select distinct  
    job_title as profession  
from  
    customer  
where  
    job_industry_category = 'Financial Services'  
    and job_title like 'Lead%'  
    and extract(year from age(current_date, dob)) > 35;
```

The results pane shows the output of the query:

Название	Источник данных
1	Senior Developer
2	Senior Sales Associate

The status bar at the bottom indicates that 2 rows were retrieved in 0.0s.

Рисунок 8 – Реализация запроса № 3

Оказалось, что клиенты из сферы Financial Services не имеют профессий, которые начинаются со слова Lead.

4. Вывести бренды, которые были куплены клиентами из сферы Financial Services, но не были куплены клиентами из сферы IT.

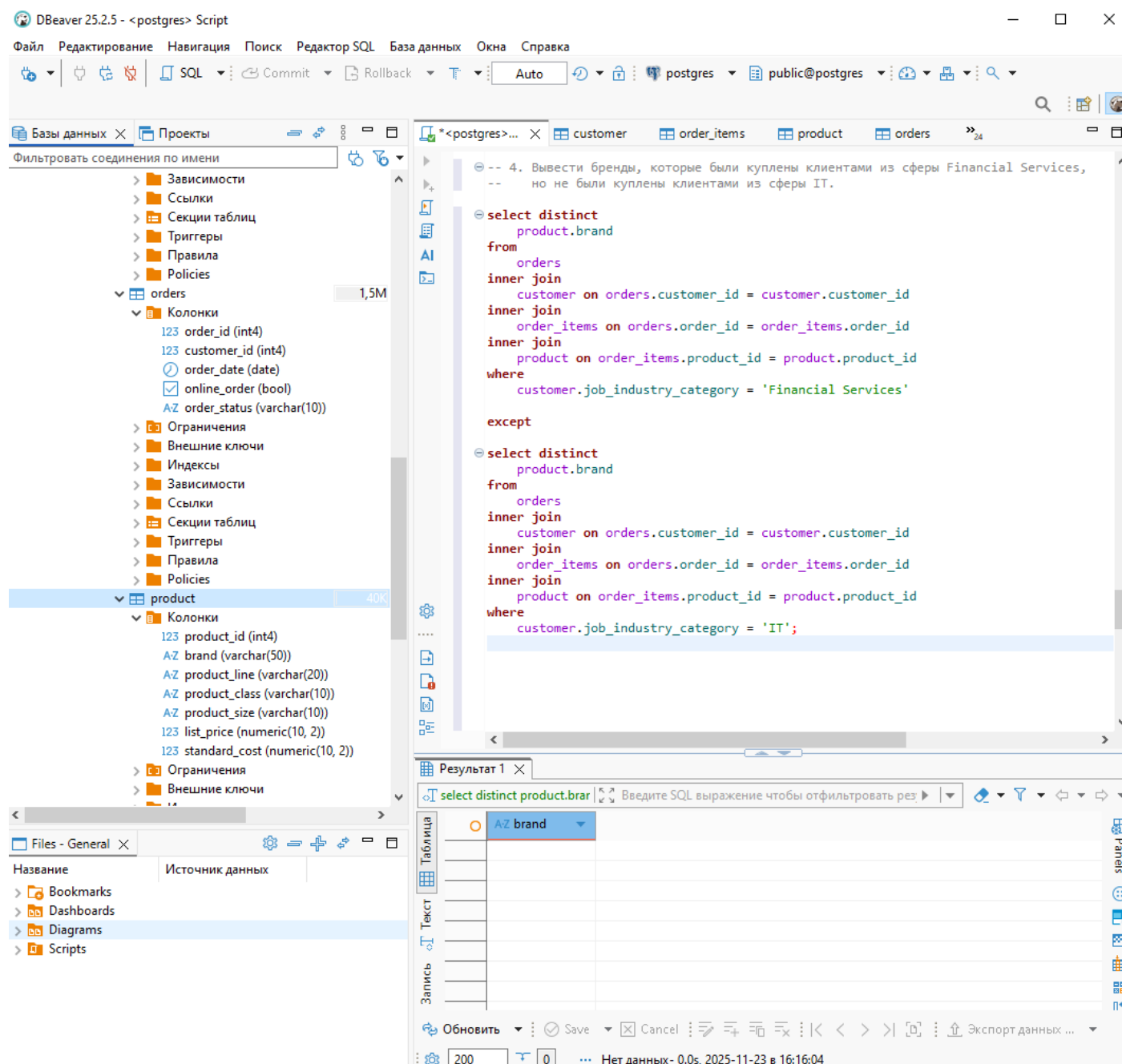


Рисунок 9 – Реализация запроса № 4 (вариант-1)

Запрос показал, что таких брендов, которые были куплены клиентами из сферы Financial Services, но не были куплены клиентами из сферы IT нет.

Представим еще один вариант этого запроса

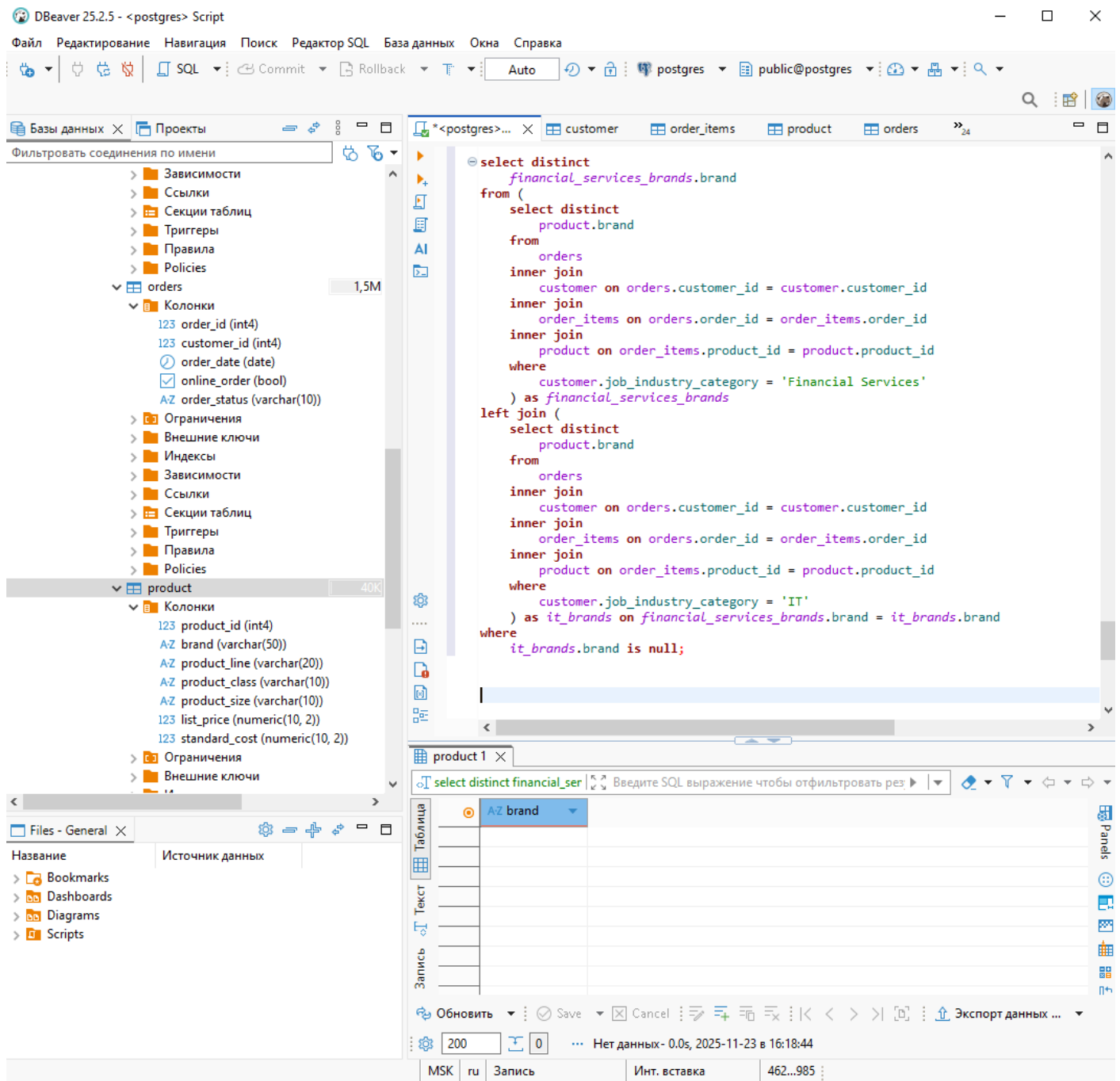


Рисунок 10 – Реализация запроса № 4 (вариант-2)

Запрос также показал, что брендов, которые были куплены клиентами из сферы Financial Services, но не были куплены клиентами из сферы IT нет.

Проверим какие бренды покупали клиенты из этих двух сфер.

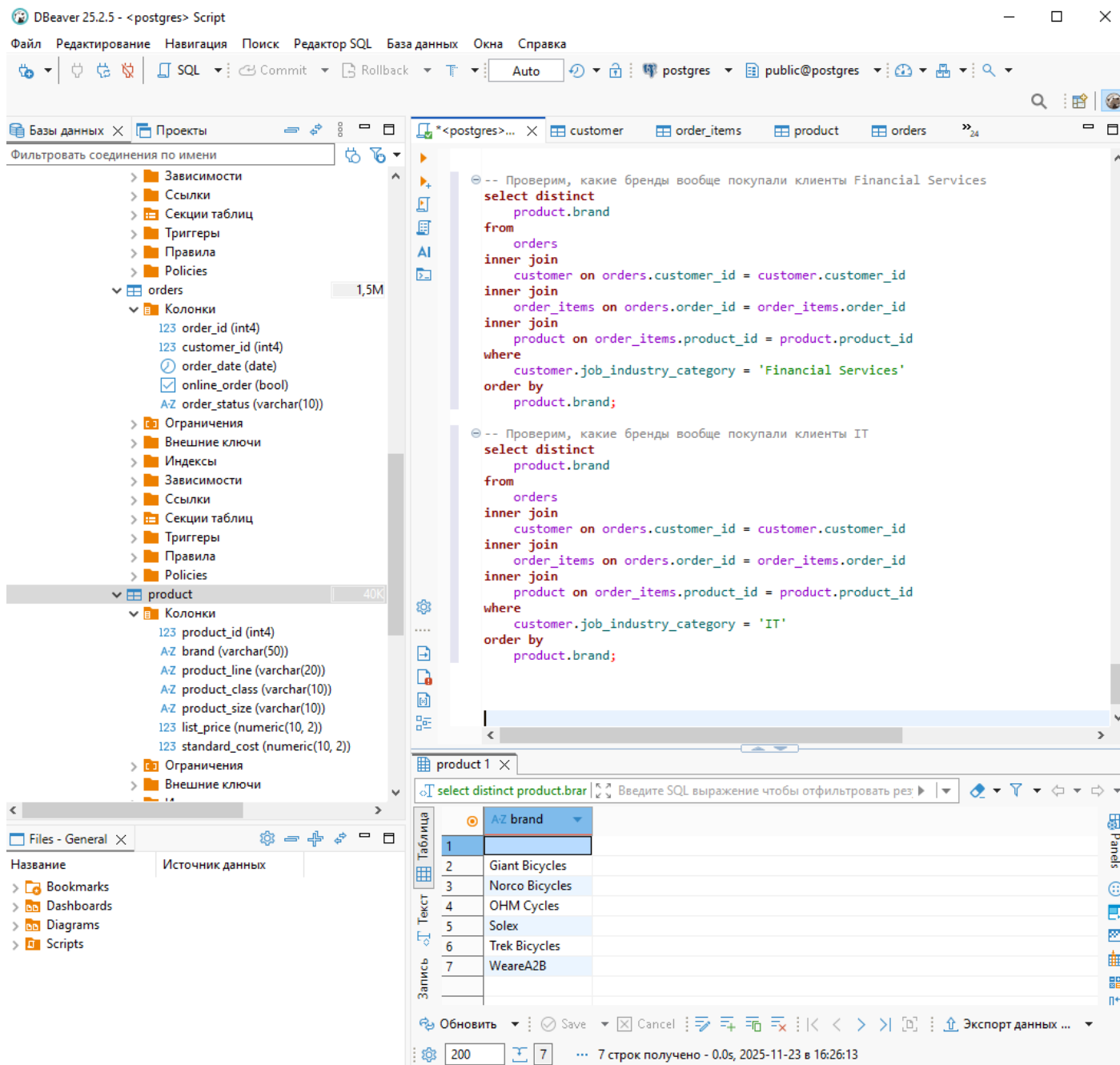


Рисунок 11 – Проверка запроса № 4

Запросы на проверку, какие бренды покупали клиенты из сферы Financial Services и из сферы IT,

показали один и тот же список брендов.

Таким образом, бренды, которые были куплены клиентами из сферы Financial Services, но не были куплены клиентами из сферы IT нет.

5. Вывести 10 клиентов (ID, имя, фамилия), которые совершили наибольшее количество онлайн-заказов (в штуках) брендов Giant Bicycles, Norco Bicycles, Trek Bicycles, при условии, что они активны и имеют оценку имущества (property_valuation) выше среднего среди клиентов из того же штата.

The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database schema for 'postgres', including tables like 'customer', 'order_items', 'product', and 'orders'. The main editor shows a SQL script for query 5. The script uses CTEs to calculate the average property valuation by state and then filters for active customers with a property valuation above the state average, ordering them by the number of online orders.

```
-- 5. Вывести 10 клиентов (ID, имя, фамилия), которые совершили
-- наибольшее количество онлайн-заказов (в штуках) брендов Giant Bicycles,
-- Norco Bicycles, Trek Bicycles, при условии,
-- что они активны и имеют оценку имущества (property_valuation)
-- выше среднего среди клиентов из того же штата.

with state_avg_property as (
    select
        state,
        avg(property_valuation) as avg_property_valuation
    from
        customer
    where
        deceased_indicator = 'N'
    group by
        state
),
filtered_customers as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.state,
        customer.property_valuation
    from
        customer
    inner join
        state_avg_property on customer.state = state_avg_property.state
    where
        customer.deceased_indicator = 'N'
        and customer.property_valuation > state_avg_property.avg_property_valuation
),
online_orders_count as (
    select
        filtered_customers.customer_id,
```

The results pane shows a table with 10 rows of customer data:

	customer_id	first_name	last_name
1	353	Antonia	Cardis
2	3 221	Brigid	Quigley
3	1 302	Ericka	Eggers
4	534	Madel	Palfrey
5	787	Norma	Batrim
6	1 117	Georgena	Guilaem
7	25	Geoff	Assaf
8	1	Laraine	Medendorp
9	478	Darcey	Harberer
10	714	Burtie	Scintsbury

The status bar at the bottom indicates that 10 rows were retrieved.

Рисунок 12 – Реализация запроса № 5

Полная версия запроса представлена ниже.

```

with state_avg_property as (
    select
        state,
        avg(property_valuation) as avg_property_valuation
    from
        customer
    where
        deceased_indicator = 'N'
    group by
        state
),
filtered_customers as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.state,
        customer.property_valuation
    from
        customer
    inner join
        state_avg_property on customer.state = state_avg_property.state
    where
        customer.deceased_indicator = 'N'
        and customer.property_valuation > state_avg_property.avg_property_valuation
),
online_orders_count as (
    select
        filtered_customers.customer_id,
        filtered_customers.first_name,
        filtered_customers.last_name,
        count(distinct orders.order_id) as online_orders_count
    from
        filtered_customers
    inner join
        orders on filtered_customers.customer_id = orders.customer_id
    inner join
        order_items on orders.order_id = order_items.order_id
    inner join
        product on order_items.product_id = product.product_id
    where
        orders.online_order = true
        and orders.order_status = 'Approved'
        and product.brand in ('Giant Bicycles', 'Norco Bicycles', 'Trek Bicycles')
    group by
        filtered_customers.customer_id,
        filtered_customers.first_name,
        filtered_customers.last_name
)
select
    customer_id,
    first_name,
    last_name
from
    online_orders_count
order by
    online_orders_count desc
limit 10;

```

Запрос можно улучшить и выводить для наглядности также количество заказов, изменив последние 9 строчек.

DBeeer 25.2.5 - <postgres> Script

Файл Редактирование Навигация Поиск Редактор SQL База данных Окна Справка

SQL Commit Rollback Auto postgres public@postgres

Базы данных Проекты

Фильтровать соединения по имени

- address 816K
- customer 752K
 - Колонки
 - 123 customer_id (int4)
 - AZ first_name (varchar(50))
 - AZ last_name (varchar(50))
 - AZ gender (varchar(10))
 - dob (date)
 - AZ job_title (varchar(100))
 - AZ job_industry_category (varchar(50))
 - AZ wealth_segment (varchar(20))
 - AZ deceased_indicator (varchar(1))
 - AZ owns_car (varchar(3))
 - AZ address (varchar(200))
 - 123 postcode (int4)
 - AZ state (varchar(20))
 - AZ country (varchar(20))
 - 123 property_valuation (int4)
 - Ограничения
 - Внешние ключи
 - Индексы
 - Зависимости
 - Ссылки
 - Секции таблиц
 - Триггеры
 - Правила
 - Политики
- customer_hw1 672K
- customer_old 752K
- job_industry 144K
- order_items 1,6M
 - Колонки
 - 123 order_item_id (int4)

```

        filtered_customers.customer_id,
        filtered_customers.first_name,
        filtered_customers.last_name,
        count(distinct orders.order_id) as online_orders_count
    from
        filtered_customers
    inner join
        orders on filtered_customers.customer_id = orders.customer_id
    inner join
        order_items on orders.order_id = order_items.order_id
    inner join
        product on order_items.product_id = product.product_id
    where
        orders.online_order = true
        and orders.order_status = 'Approved'
        and product.brand in ('Giant Bicycles', 'Norco Bicycles', 'Trek Bicycles')
    group by
        filtered_customers.customer_id,
        filtered_customers.first_name,
        filtered_customers.last_name
    )
    select
        customer_id,
        first_name,
        last_name,
        online_orders_count
    from
        online_orders_count
    order by
        online_orders_count desc
    limit 10;

```

customer 1

with state_avg_property as

	123 customer_id	AZ first_name	AZ last_name	123 online_orders_count
1	353	Antonia	Cardis	7
2	3 221	Brigid	Quigley	7
3	1 302	Ericka	Eggers	6
4	534	Madel	Palfrey	6
5	787	Norma	Batrim	6
6	1 117	Georgena	Guilaem	6
7	25	Geoff	Assaf	6
8	1	Laraine	Medendorp	6
9	478	Darcey	Harberer	6
10	714	Burtie	Scintsbury	6

Обновить Save Cancel Экспорт данных ...

200 10 10 строк получено - 0.0s, 2025-11-23 в 17:31:02

Рисунок 13 – Реализация запроса № 5 (с выводом количества заказов)

6. Вывести всех клиентов (ID, имя, фамилия), у которых нет подтвержденных онлайн-заказов за последний год, но при этом они владеют автомобилем и их сегмент благосостояния не Mass Customer.

The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database schema for 'postgres', including tables like 'customer', 'order_items', 'product', and 'orders'. The main editor shows a SQL script with a comment in Russian and a query to select customers who own a car, are not 'Mass Customer', and have no confirmed online orders in the last year. The bottom pane shows the results of the query in a table format.

```
-- 6. Вывести всех клиентов (ID, имя, фамилия),
-- у которых нет подтвержденных онлайн-заказов за последний год,
-- но при этом они владеют автомобилем
-- и их сегмент благосостояния не Mass Customer.

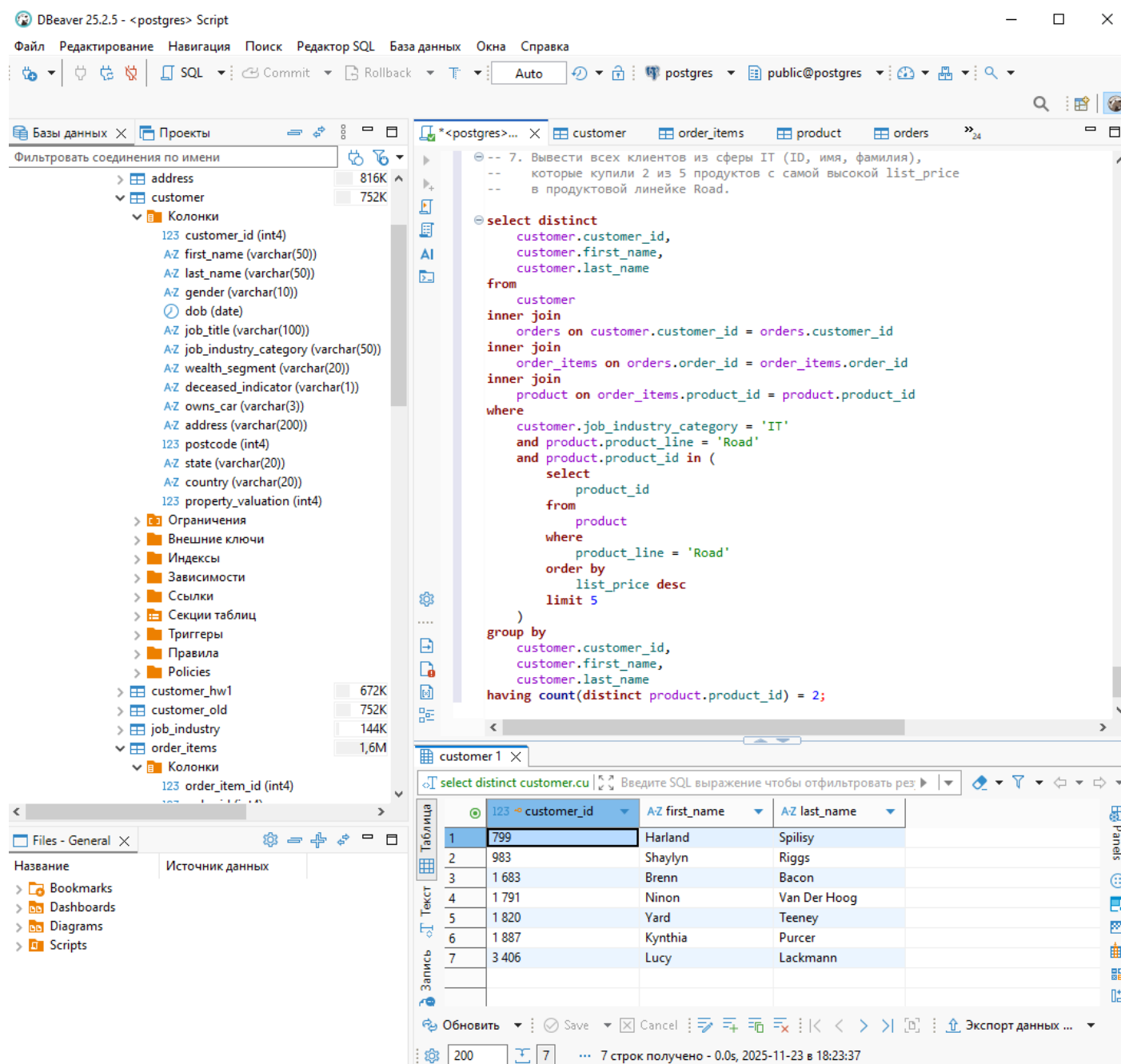
select
  customer_id,
  first_name,
  last_name
from
  customer
where
  owns_car = 'Yes'
  and wealth_segment != 'Mass Customer'
  and customer_id not in (
    select distinct
      customer_id
    from
      orders
    where
      online_order = true
      and order_status = 'Approved'
      and order_date >= current_date - interval '1 year'
  );
```

	customer_id	first_name	last_name
1	5	Sheila-kathryn	Calton
2	6	Curr	Duckhouse
3	7	Fina	Merali
4	9	Mala	Lind
5	13	Gabriele	Norcross
6	17	Heath	Faraday
7	21	Mile	Cammocke
8	23	Olav	Polak
9	28	Fee	Zellmer
10	30	Darrick	Helleckas
11	31	Star	Praton

Рисунок 14 – Реализация запроса № 6

По запросу было выведено достаточно много клиентов. Это связано с тем, что заказы, в основном, датируются 2017 годом.

7. Вывести всех клиентов из сферы IT (ID, имя, фамилия), которые купили 2 из 5 продуктов с самой высокой list_price в продуктовой линейке Road.



The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database schema, including tables like 'customer', 'order_items', and 'product'. The main editor shows a SQL script for query 7. The script selects distinct customer information (ID, first name, last name) from the 'customer' table, joined with 'orders' and 'order_items', and filtered by 'product' table. The filter conditions are: customer.job_industry_category = 'IT', product.product_line = 'Road', and product.product_id in a subquery that selects the top 5 most expensive products from the 'Road' line. The results are grouped by customer information and filtered by having at least 2 distinct product IDs. The bottom panel shows the results of the query, displaying 7 rows of customer data.

customer_id	first_name	last_name
799	Harland	Spilisy
983	Shaylyn	Riggs
1 683	Brenn	Bacon
1 791	Ninon	Van Der Hoog
1 820	Yard	Teeney
1 887	Kynthia	Purcer
3 406	Lucy	Lackmann

Рисунок 15 – Реализация запроса № 7

Недостатком такого варианта запроса является то, что он не учитывает тех клиентов, которые купили более 2 товаров из топ-5 с высокой list_price. Поэтому логичнее было бы заменить последнюю строку

```
having count(distinct product.product_id) = 2;
```

на

```
having count(distinct product.product_id) >= 2;
```


The screenshot shows the DBeaver 25.2.5 interface. The top menu bar includes options like 'Файл', 'Редактирование', 'Навигация', 'Поиск', 'Редактор SQL', 'База данных', 'Окна', and 'Справка'. The left sidebar displays a database schema for 'customer' with columns like 'customer_id', 'first_name', 'last_name', etc. The main editor window contains a SQL script with a complex query involving multiple joins and a subquery. The bottom panel shows the results of the query in a table format.

SQL Query:

```
-- 7. Вывести всех клиентов из сферы IT (ID, имя, фамилия),
-- которые купили 2 из 5 продуктов с самой высокой list_price
-- в продуктовой линейке Road.

select distinct
customer.customer_id,
customer.first_name,
customer.last_name
from
customer
inner join
orders on customer.customer_id = orders.customer_id
inner join
order_items on orders.order_id = order_items.order_id
inner join
product on order_items.product_id = product.product_id
where
customer.job_industry_category = 'IT'
and product.product_line = 'Road'
and product.product_id in (
select
product_id
from
product
where
product_line = 'Road'
order by
list_price desc
limit 5
)
group by
customer.customer_id,
customer.first_name,
customer.last_name
having count(distinct product.product_id) >= 2;
```

Results Table:

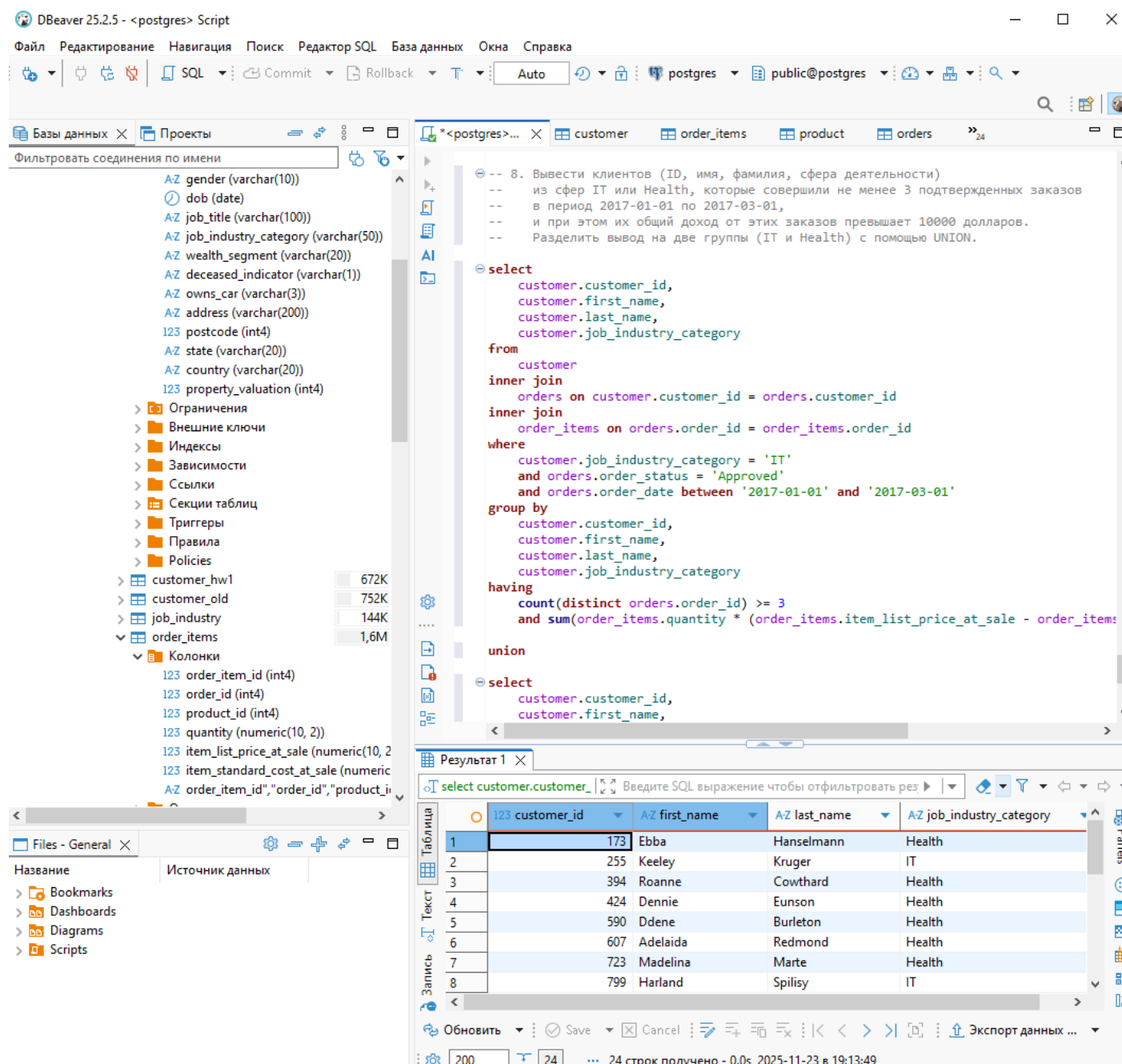
customer_id	first_name	last_name
799	Harland	Spilisy
983	Shaylyn	Riggs
1 683	Brenn	Bacon
1 791	Ninon	Van Der Hoog
1 820	Yard	Teeney
1 887	Kynthia	Purcer
3 406	Lucy	Lackmann

Рисунок 16 – Реализация запроса № 7 (модификация запроса)

Следует отметить, что модифицированный запрос дает тот же результат. Это означает, что более 2 товаров из топ-5 с самой высокой list_price клиенты не покупали.

8. Вывести клиентов (ID, имя, фамилия, сфера деятельности) из сфер IT или Health, которые совершили не менее 3 подтвержденных заказов в период 2017-01-01 по 2017-03-01, и при этом их общий доход от этих заказов превышает 10 000 долларов. Разделить вывод на две группы (IT и Health) с помощью UNION.

Будем считать, что доход клиенту приносит разница между ценой продажи и стандартной стоимостью товара. В таблице **order_items** **item_list_price_at_sale** превышает **item_standard_cost_at_sale**, тогда доход клиента считается как $quantity * (item_list_price_at_sale - item_standard_cost)$.



The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database schema for 'customer' and 'order_items'. The main editor contains the following SQL script:

```
-- 8. Вывести клиентов (ID, имя, фамилия, сфера деятельности)
-- из сфер IT или Health, которые совершили не менее 3 подтвержденных заказов
-- в период 2017-01-01 по 2017-03-01,
-- и при этом их общий доход от этих заказов превышает 10000 долларов.
-- Разделить вывод на две группы (IT и Health) с помощью UNION.

select
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    customer.job_industry_category
from
    customer
inner join
    orders on customer.customer_id = orders.customer_id
inner join
    order_items on orders.order_id = order_items.order_id
where
    customer.job_industry_category = 'IT'
    and orders.order_status = 'Approved'
    and orders.order_date between '2017-01-01' and '2017-03-01'
group by
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    customer.job_industry_category
having
    count(distinct orders.order_id) >= 3
    and sum(order_items.quantity * (order_items.item_list_price_at_sale - order_items.item_standard_cost_at_sale)) > 10000
union
select
    customer.customer_id,
    customer.first_name,
```

The results table shows the following data:

	customer_id	first_name	last_name	job_industry_category
1	173	Ebba	Hanselmann	Health
2	255	Keeley	Kruger	IT
3	394	Roanne	Cowthard	Health
4	424	Dennie	Eunson	Health
5	590	Ddene	Burleton	Health
6	607	Adelaida	Redmond	Health
7	723	Madelina	Marte	Health
8	799	Harland	Spilis	IT

Рисунок 17 – Реализация запроса № 8

Полная версия запроса представлена ниже.

```

select
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    customer.job_industry_category
from
    customer
inner join
    orders on customer.customer_id = orders.customer_id
inner join
    order_items on orders.order_id = order_items.order_id
where
    customer.job_industry_category = 'IT'
    and orders.order_status = 'Approved'
    and orders.order_date between '2017-01-01' and '2017-03-01'
group by
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    customer.job_industry_category
having
    count(distinct orders.order_id) >= 3
    and sum(order_items.quantity * (order_items.item_list_price_at_sale -
order_items.item_standard_cost_at_sale)) > 10000

union

select
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    customer.job_industry_category
from
    customer
inner join
    orders on customer.customer_id = orders.customer_id
inner join
    order_items on orders.order_id = order_items.order_id
where
    customer.job_industry_category = 'Health'
    and orders.order_status = 'Approved'
    and orders.order_date between '2017-01-01' and '2017-03-01'
group by
    customer.customer_id,
    customer.first_name,
    customer.last_name,
    customer.job_industry_category
having
    count(distinct orders.order_id) >= 3
    and sum(order_items.quantity * (order_items.item_list_price_at_sale -
order_items.item_standard_cost_at_sale)) > 10000

order by
    customer_id;

```