

Домашнее задание 3. Группировка данных и оконные функции

Даны CSV-файлы:

customer.csv

product.csv

orders.csv

order_items.csv

Предварительный экспресс анализ данных выполненный с помощью pandas, показал, что все таблицы идентичны таблицам из предыдущего домашнего задания, поэтому создание новых таблиц не требуется. Результаты сохранены в файле pandas_analysis_3.ipynb.

Оставим описание данных таблиц для наглядности.

Таблица **customer**:

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

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)

Выполнение запросов

1. Вывести распределение (количество) клиентов по сферам деятельности, отсортировав результат по убыванию количества.

The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database structure for 'postgres' on 'localhost:5432', with the 'public' schema selected. The 'customer' table is highlighted, showing columns like 'customer_id', 'first_name', 'last_name', etc. The main editor displays three SQL queries for 'job_industry_category' distribution. The bottom panel shows the results of the first query, sorted by 'cnt_customers' in descending order.

SQL Queries:

```
-- ДЗ 3
-- 1. Вывести распределение (количество) клиентов по сферам деятельности,
-- отсортировав результат по убыванию количества.

-- вариант 1
select
    job_industry_category,
    count(*) as cnt_customers
from
    customer
group by
    job_industry_category
order by
    cnt_customers desc;

-- вариант 2
select
    job_industry_category,
    count(customer_id) as cnt_customers
from
    customer
group by
    job_industry_category
order by
    cnt_customers desc;

-- вариант 3
select distinct
    job_industry_category,
    (count(*) over(partition by job_industry_category)) as cnt_customers
from
    customer
order by
    cnt_customers desc;
```

Query Results (Table: customer1):

job_industry_category	cnt_customers
Manufacturing	799
Financial Services	774
n/a	656
Health	602
Retail	358
Property	267

Рисунок 1 – Реализация запроса № 1

2. Найти общую сумму дохода ($list_price * quantity$) по всем подтвержденным заказам за каждый месяц по сферам деятельности клиентов. Отсортировать результат по году, месяцу и сфере деятельности.

Следует отметить что `list_price` в таблице **product** совпадает с `item_list_price_at_sale` в таблице **order_items**.

The screenshot shows the DBeaver 25.2.5 interface with a PostgreSQL database. The SQL editor contains a query to calculate the total income by year, month, and industry category for approved orders. The results are displayed in a table with 4 columns: year, month, job_industry_category, and total_income.

```
-- 2. Найти общую сумму дохода (list_price*quantity)
-- по всем подтвержденным заказам за каждый месяц по сферам
-- деятельности клиентов. Отсортировать результат по году,
-- месяцу и сфере деятельности.

select
    extract(year from orders.order_date) as year,
    extract(month from orders.order_date) as month,
    customer.job_industry_category,
    sum(order_items.item_list_price_at_sale * order_items.quantity) as total_income
from
    orders
inner join
    customer on orders.customer_id = customer.customer_id
inner join
    order_items on orders.order_id = order_items.order_id
where
    orders.order_status = 'Approved'
group by
    extract(year from orders.order_date),
    extract(month from orders.order_date),
    customer.job_industry_category
order by
    year,
    month,
    customer.job_industry_category;
```

	123 year	123 month	AZ job_industry_category	123 total_income
1	2017		Argiculture	232 148,25
2	2017		Entertainment	342 541,17
3	2017		Financial Services	2 032 708,45
4	2017		Health	1 570 012,48
5	2017		IT	604 949,53
6	2017		Manufacturing	1 931 238,45
7	2017		n/a	1 788 848,1

Обновить | Save | Cancel | 200 | 120 | 120 строк получено - 0.0s, 2025-11-30 в 22:02:33

MSK | ru | Запись | Инт. вставка | 73

Рисунок 2 – Реализация запроса № 2
(код без подсветки почему-то, но рабочий)

3. Вывести количество уникальных онлайн-заказов для всех брендов в рамках подтвержденных заказов клиентов из сферы IT. Включить бренды, у которых нет онлайн-заказов от IT-клиентов, — для них должно быть указано количество 0.

The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays a database schema with tables like `customer_hw1`, `customer_old`, `job_industry`, `order_items`, and `orders`. The main editor shows a SQL script for query 3, which counts unique online orders for IT clients. The script is as follows:

```
-- 3. Вывести количество уникальных онлайн-заказов для всех брендов
-- в рамках подтвержденных заказов клиентов из сферы IT.
-- Включить бренды, у которых нет онлайн-заказов от IT-клиентов,
-- для них должно быть указано количество 0.

select
    product.brand,
    count(distinct case
        when orders.online_order = true
            and customer.job_industry_category = 'IT'
            and orders.order_status = 'Approved'
        then orders.order_id
    end) as unique_online_orders
from
    product
left join
    order_items on product.product_id = order_items.product_id
left join
    orders on order_items.order_id = orders.order_id
left join
    customer on orders.customer_id = customer.customer_id
group by
    product.brand
order by
    unique_online_orders desc;
```

Below the script, the results are displayed in a table with columns `AZ brand` and `unique_online_orders`:

AZ brand	unique_online_orders
Solex	202
Giant Bicycles	194
OHM Cycles	174
Norco Bicycles	173
WeareA2B	171
Trek Bicycles	160
	44

The bottom status bar indicates that 7 rows were retrieved on 2025-11-30 at 16:51:59.

Рисунок 3 – Реализация запроса № 3

4. Найти по всем клиентам: сумму всех заказов (общего дохода), максимум, минимум и количество заказов, а также среднюю сумму заказа по каждому клиенту. Отсортировать результат по убыванию суммы всех заказов и количества заказов. Выполнить двумя способами: используя только GROUP BY и используя только оконные функции. Сравнить результат.

The screenshot shows the DBeaver 25.2.5 interface. The top menu bar includes 'Файл', 'Редактирование', 'Навигация', 'Поиск', 'Редактор SQL', 'База данных', 'Окна', and 'Справка'. The toolbar contains icons for various database operations. The left sidebar shows a tree view of the database schema, with 'customer' selected. The main editor displays a SQL script for task 4, which calculates total income, maximum, minimum, and average order value for each customer, sorted by total income and order count. The script uses GROUP BY and COALESCE functions. The bottom panel shows the results of the query in a table view, with columns for customer_id, first_name, last_name, total_income, and count_order. The table contains 6 rows of data.

```
-- 4. Найти по всем клиентам: сумму всех заказов (общего дохода),
-- максимум, минимум и количество заказов,
-- а также среднюю сумму заказа по каждому клиенту.
-- Отсортировать результат по убыванию суммы всех заказов
-- и количества заказов. Выполнить двумя способами:
-- используя только GROUP BY и используя только оконные функции.
-- Сравнить результат.

-- вариант 1

select
customer.customer_id,
customer.first_name,
customer.last_name,
coalesce(sum(order_items.item_list_price_at_sale * order_items.quantity), 0)
as total_income,
coalesce(max(order_items.item_list_price_at_sale * order_items.quantity), 0)
as max_order_value,
coalesce(min(order_items.item_list_price_at_sale * order_items.quantity), 0)
as min_order_value,
coalesce(count(orders.order_id), 0) as count_order,
coalesce(avg(order_items.item_list_price_at_sale * order_items.quantity), 0)
as avg_order_value
from
customer
left join
orders on customer.customer_id = orders.customer_id
left join
order_items on orders.order_id = order_items.order_id
group by
customer.customer_id,
customer.first_name,
customer.last_name
order by
total_income desc,
count_order desc;
```

	customer_id	first_name	last_name	total_income	count_order
1	2 183	Jillie	Fyndon	136 632,46	12
2	1 597	Jeffry	Slowly	133 657,06	12
3	941	Tye	Doohan	129 789,94	12
4	1 129	Hercule	Doohan	129 189,48	12
5	637	Mercy	Wilsone	109 334,74	12
6	2 309	Herc	McIlhone	107 476,68	12

Рисунок 4 – Реализация запроса № 4
(с использованием GROUP BY)

The screenshot shows the DBeaver 25.2.5 interface. The top menu bar includes 'Файл', 'Редактирование', 'Навигация', 'Поиск', 'Редактор SQL', 'База данных', 'Окна', and 'Справка'. The toolbar contains icons for various database actions. The left sidebar shows a tree view of the database schema, including 'address' (816K) and 'customer' (752K) tables. The main editor displays a SQL script titled '-- вариант 2' with the following query:

```
-- вариант 2
select distinct
customer.customer_id,
customer.first_name,
customer.last_name,
coalesce(sum(order_items.item_list_price_at_sale * order_items.quantity)
over (partition by customer.customer_id), 0) as total_income,
coalesce(max(order_items.item_list_price_at_sale * order_items.quantity)
over (partition by customer.customer_id), 0) as max_order_value,
coalesce(min(order_items.item_list_price_at_sale * order_items.quantity)
over (partition by customer.customer_id), 0) as min_order_value,
coalesce(count(order_items.order_id)
over (partition by customer.customer_id), 0) as count_order,
coalesce(avg(order_items.item_list_price_at_sale * order_items.quantity)
over (partition by customer.customer_id), 0) as avg_order_value
from
customer
left join
orders on customer.customer_id = orders.customer_id
left join
order_items on orders.order_id = order_items.order_id
order by
total_income desc,
count_order desc;
```

The bottom panel shows the results of the query in a table view. The table has columns: 'customer_id', 'first_name', 'last_name', 'total_income', and 'max_order_value'. The data is as follows:

	customer_id	first_name	last_name	total_income	max_order_value
1	2183	Jillie	Fyndon	136 632,46	
2	1597	Jeffry	Slowly	133 657,06	
3	941	Tye	Doohan	129 789,94	
4	1129	Hercule		129 189,48	
5	637	Mercy	Wilsone	109 334,74	
6	2309	Herc	McIlhone	107 476,68	

The bottom status bar indicates '200 строк получено - 0.0s, 2025-11-30 в 22:09:22'.

Рисунок 5 – Реализация запроса № 4
(с использованием оконных функций)

Следует отметить, что использование **select distinct** позволило в варианте 2 с оконными функциями получить тот же результат, что и в варианте 1. Запрос возвращает по одной строке на клиента. Просто **select** в варианте 2 возвращает много строк для одного клиента.

Также возможен вопрос к решению, связанный с уникальностью **order_id**. Уникальность была предварительно проверена и **order_id** в таблице **orders** primary key. Можно дополнительно подтвердить уникальность запросом.

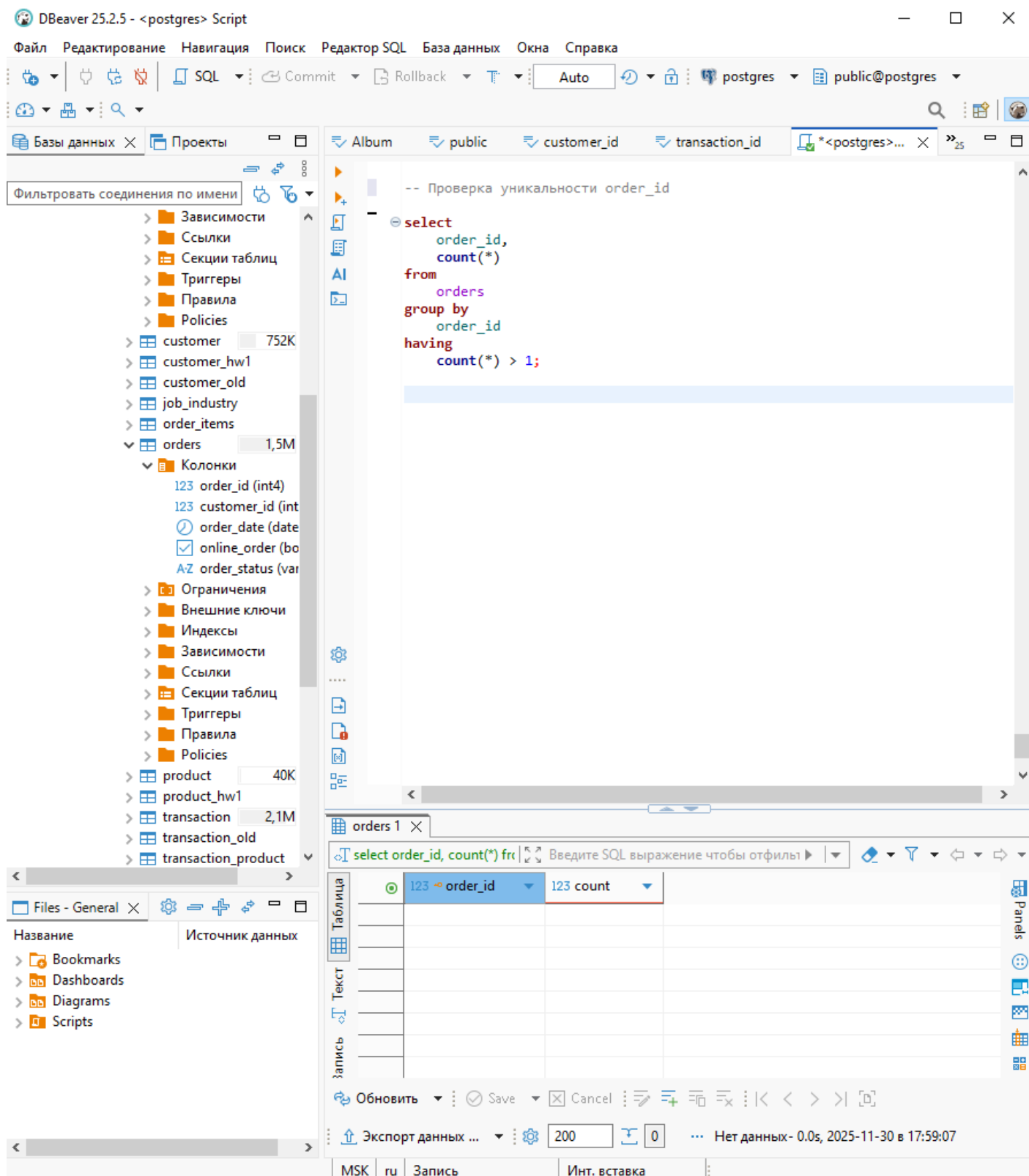
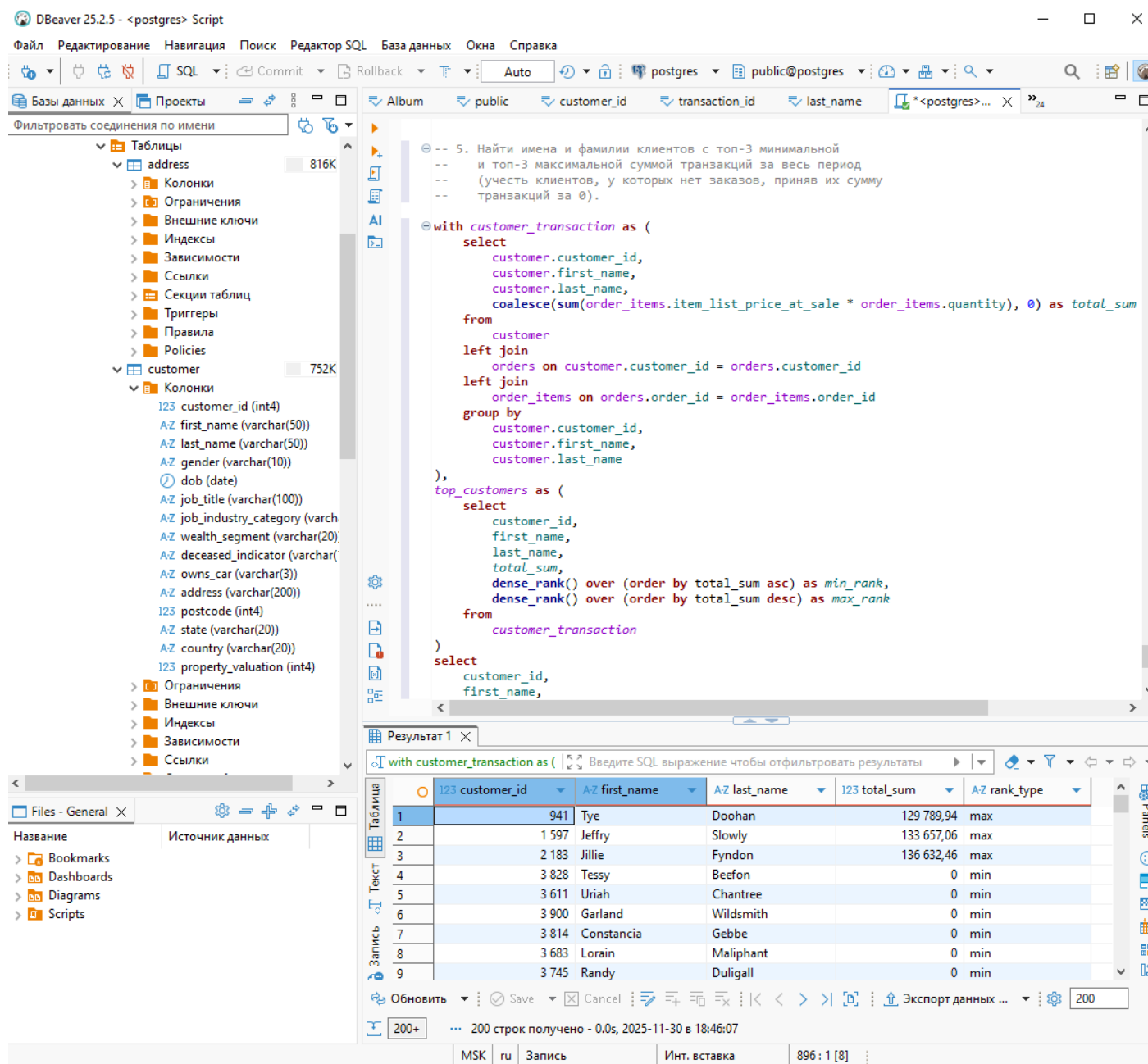


Рисунок 6 – Дополнительная проверка уникальности order_id для запроса № 4

5. Найти имена и фамилии клиентов с топ-3 минимальной и топ-3 максимальной суммой транзакций за весь период (учесть клиентов, у которых нет заказов, приняв их сумму транзакций за 0).



The screenshot displays the DBeaver 25.2.5 interface with a PostgreSQL database. The SQL editor contains the following query:

```
-- 5. Найти имена и фамилии клиентов с топ-3 минимальной
-- и топ-3 максимальной суммой транзакций за весь период
-- (учесть клиентов, у которых нет заказов, приняв их сумму
-- транзакций за 0).

with customer_transaction as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        coalesce(sum(order_items.item_list_price_at_sale * order_items.quantity), 0) as total_sum
    from
        customer
    left join
        orders on customer.customer_id = orders.customer_id
    left join
        order_items on orders.order_id = order_items.order_id
    group by
        customer.customer_id,
        customer.first_name,
        customer.last_name
),
top_customers as (
    select
        customer_id,
        first_name,
        last_name,
        total_sum,
        dense_rank() over (order by total_sum asc) as min_rank,
        dense_rank() over (order by total_sum desc) as max_rank
    from
        customer_transaction
)
select
    customer_id,
    first_name,
```

The results are displayed in a table with the following columns: 123 customer_id, AZ first_name, AZ last_name, 123 total_sum, and AZ rank_type. The table contains 9 rows of data:

	123 customer_id	AZ first_name	AZ last_name	123 total_sum	AZ rank_type
1	941	Tye	Doohan	129 789,94	max
2	1 597	Jeffry	Slowly	133 657,06	max
3	2 183	Jillie	Fyndon	136 632,46	max
4	3 828	Tessy	Beefon	0	min
5	3 611	Uriah	Chantree	0	min
6	3 900	Garland	Wildsmith	0	min
7	3 814	Constancia	Gebbe	0	min
8	3 683	Lorain	Maliphant	0	min
9	3 745	Randy	Duligall	0	min

Рисунок 7 – Реализация запроса № 5

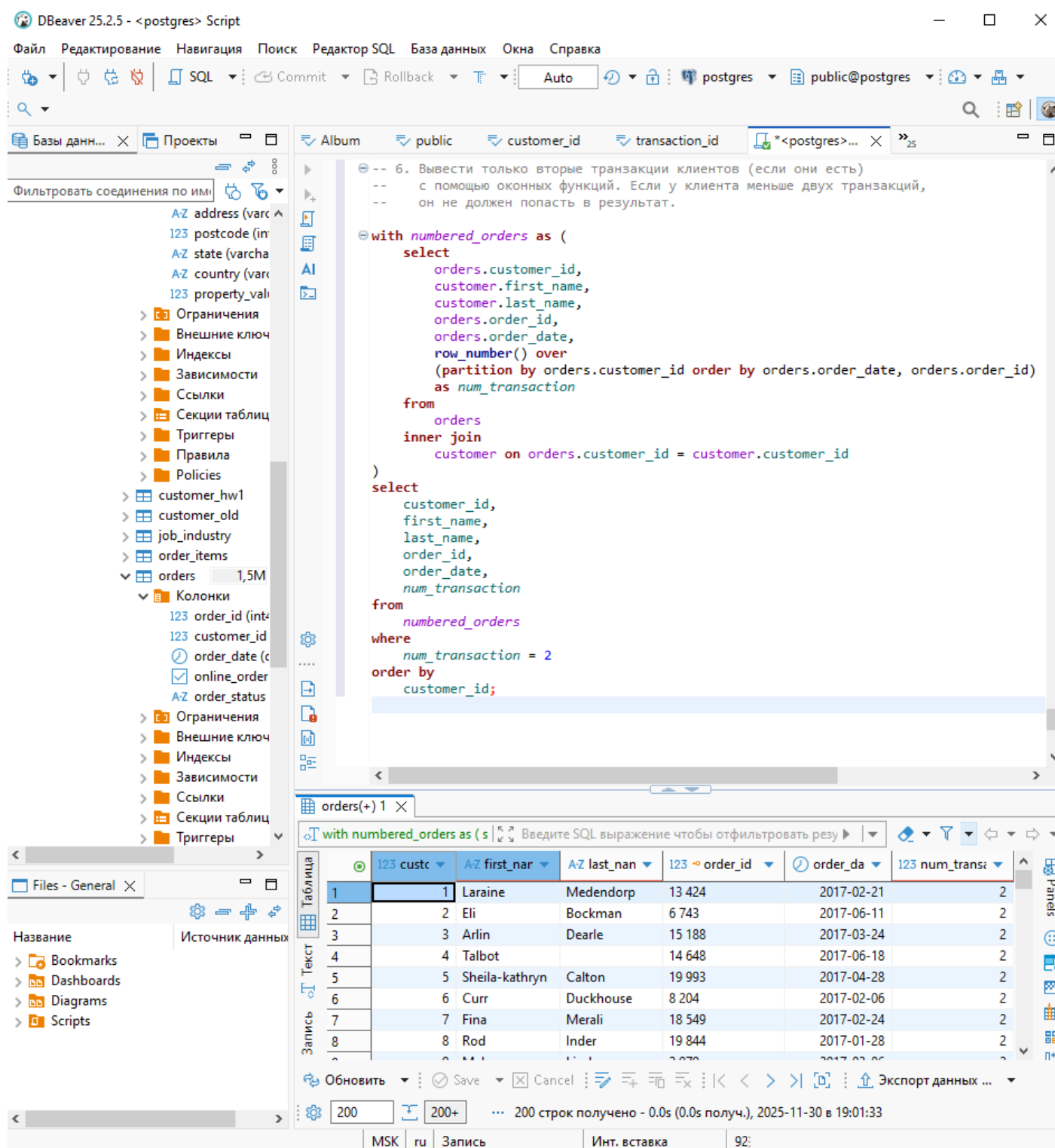
Полный код запроса представлен ниже.

```

with customer_transaction as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        coalesce(sum(order_items.item_list_price_at_sale * order_items.quantity), 0) as
total_sum
    from
        customer
    left join
        orders on customer.customer_id = orders.customer_id
    left join
        order_items on orders.order_id = order_items.order_id
    group by
        customer.customer_id,
        customer.first_name,
        customer.last_name
),
top_customers as (
    select
        customer_id,
        first_name,
        last_name,
        total_sum,
        dense_rank() over (order by total_sum asc) as min_rank,
        dense_rank() over (order by total_sum desc) as max_rank
    from
        customer_transaction
)
select
    customer_id,
    first_name,
    last_name,
    total_sum,
    'min' as rank_type
from
    top_customers
where
    min_rank <= 3
union all
select
    customer_id,
    first_name,
    last_name,
    total_sum,
    'max' as rank_type
from
    top_customers
where
    max_rank <= 3
order by
    rank_type,
    total_sum;

```

6. Вывести только вторые транзакции клиентов (если они есть) с помощью оконных функций. Если у клиента меньше двух транзакций, он не должен попасть в результат.



The screenshot displays the DBeaver 25.2.5 interface. The left sidebar shows the database schema for 'public', including tables like 'customer' and 'orders'. The main editor shows a SQL script with a comment in Russian and a query using a window function to select the second transaction for each customer. The results pane at the bottom shows a table with 8 rows of data.

SQL Script:

```
-- 6. Вывести только вторые транзакции клиентов (если они есть)
-- с помощью оконных функций. Если у клиента меньше двух транзакций,
-- он не должен попасть в результат.

with numbered_orders as (
  select
    orders.customer_id,
    customer.first_name,
    customer.last_name,
    orders.order_id,
    orders.order_date,
    row_number() over
      (partition by orders.customer_id order by orders.order_date, orders.order_id)
      as num_transaction
  from
    orders
  inner join
    customer on orders.customer_id = customer.customer_id
)
select
  customer_id,
  first_name,
  last_name,
  order_id,
  order_date,
  num_transaction
from
  numbered_orders
where
  num_transaction = 2
order by
  customer_id;
```

Results Table:

	123 custc	AZ first_nar	AZ last_nar	123 order_id	order_da	123 num_trans
1	1	Laraine	Medendorp	13 424	2017-02-21	2
2	2	Eli	Bockman	6 743	2017-06-11	2
3	3	Arlin	Dearle	15 188	2017-03-24	2
4	4	Talbot		14 648	2017-06-18	2
5	5	Sheila-kathryn	Calton	19 993	2017-04-28	2
6	6	Curr	Duckhouse	8 204	2017-02-06	2
7	7	Fina	Merali	18 549	2017-02-24	2
8	8	Rod	Inder	19 844	2017-01-28	2

200 строк получено - 0.0s (0.0s получ.), 2025-11-30 в 19:01:33

Рисунок 8 – Реализация запроса № 6

7. Вывести имена, фамилии и профессии клиентов, а также длительность максимального интервала (в днях) между двумя последовательными заказами. Исключить клиентов, у которых только один или меньше заказов.

The screenshot shows the DBeaver 25.2.5 interface. The SQL editor contains the following query:

```
-- 7. Вывести имена, фамилии и профессии клиентов,
-- а также длительность максимального интервала (в днях)
-- между двумя последовательными заказами. Исключить клиентов,
-- у которых только один или меньше заказов.

with order_d_days as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.job_title,
        orders.order_date - lag(orders.order_date) over
            (partition by customer.customer_id order by orders.order_date)
            as days_between,
        count(*) over (partition by customer.customer_id) as total_orders
    from
        customer
    inner join
        orders on customer.customer_id = orders.customer_id
)
select distinct
    customer_id,
    first_name,
    last_name,
    job_title,
    max(days_between) over (partition by customer_id) as max_interval_days
from
    order_d_days
where
    days_between is not null
order by
    max_interval_days desc;
```

The results table shows the following data:

	123 customer_id	AZ first_name	AZ last_name	AZ job_title	123 max_interval_days
1	1 584	Susanetta		Legal Assistant	357
2	1 810	Royall	Terris	Geological Engine	330
3	2 128	Gregorius	Cockram	Data Coordinator	330
4	3 316	Stoddard	Giacomoni	Structural Analysis	330
5	3 156	Bearnard	Letixier		329
6	3 222	Caralie	Sellors	Senior Editor	321
7	335	Debee	Martynov	Senior Editor	320
8	316	Genni	Larway	Environmental Sp	314

The status bar at the bottom indicates: 200 строк получено - 0.0s, 2025-11-30 в 19:48:20.

Рисунок 9 – Реализация запроса № 7

8. Найти топ-5 клиентов (по общему доходу) в каждом сегменте благосостояния (wealth_segment). Вывести имя, фамилию, сегмент и общий доход. Если в сегменте менее 5 клиентов, вывести всех.

The screenshot shows the DBeaver 25.2.5 interface. The left sidebar displays the database schema for 'public', including tables like 'address' and 'customer'. The main editor shows a SQL query designed to find the top 5 customers by total income within each wealth segment. The query uses a CTE 'customer_income' to calculate total income and a window function 'ranked_customers' to rank them. The results pane at the bottom shows a table with 6 rows, displaying the top 5 customers for each segment, with the 6th row being a placeholder for segments with fewer than 5 customers.

```
-- 8. Найти топ-5 клиентов (по общему доходу) в каждом сегменте благосостояния
-- (wealth_segment). Вывести имя, фамилию, сегмент и общий доход.
-- Если в сегменте менее 5 клиентов, вывести всех.

with customer_income as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.wealth_segment,
        coalesce(sum(order_items.item_list_price_at_sale * order_items.quantity), 0)
        as total_income
    from
        customer
    left join
        orders on customer.customer_id = orders.customer_id
    left join
        order_items on orders.order_id = order_items.order_id
    group by
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.wealth_segment
),
ranked_customers as (
    select
        customer_id,
        first_name,
        last_name,
        wealth_segment,
        total_income,
        row_number() over (partition by wealth_segment order by total_income desc)
        as rank_in_segment
    from
        customer_income
)
select
```

	customer_id	first_name	last_name	wealth_segment	total_income
1	1 597	Jeffrey	Slowly	Affluent Customer	133 657,06
2	941	Tye	Dooohan	Affluent Customer	129 789,94
3	2 309	Herc	McIlhone	Affluent Customer	107 476,68
4	3 015	Queenie	Flips	Affluent Customer	106 182,33
5	2 914	Jessamine	Brazear	Affluent Customer	98 618,77
6	637	Mercy	Wilsone	High Net Worth	109 334,74

Рисунок 10 – Реализация запроса № 8

Полный код запроса представлен ниже.

```

with customer_income as (
    select
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.wealth_segment,
        coalesce(sum(order_items.item_list_price_at_sale * order_items.quantity), 0)
        as total_income
    from
        customer
    left join
        orders on customer.customer_id = orders.customer_id
    left join
        order_items on orders.order_id = order_items.order_id
    group by
        customer.customer_id,
        customer.first_name,
        customer.last_name,
        customer.wealth_segment
),
ranked_customers as (
    select
        customer_id,
        first_name,
        last_name,
        wealth_segment,
        total_income,
        row_number() over (partition by wealth_segment order by total_income desc)
        as rank_in_segment
    from
        customer_income
)
select
    customer_id,
    first_name,
    last_name,
    wealth_segment,
    total_income,
    rank_in_segment
from
    ranked_customers
where
    rank_in_segment <= 5
order by
    wealth_segment,
    rank_in_segment;

```