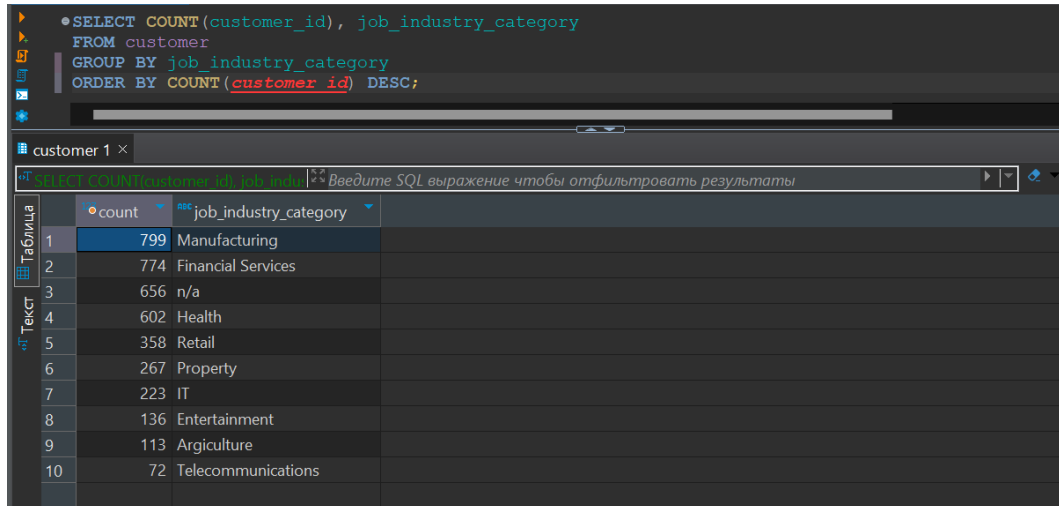


- Вывести распределение (количество) клиентов по сферам деятельности, отсортировав результат по убыванию количества.

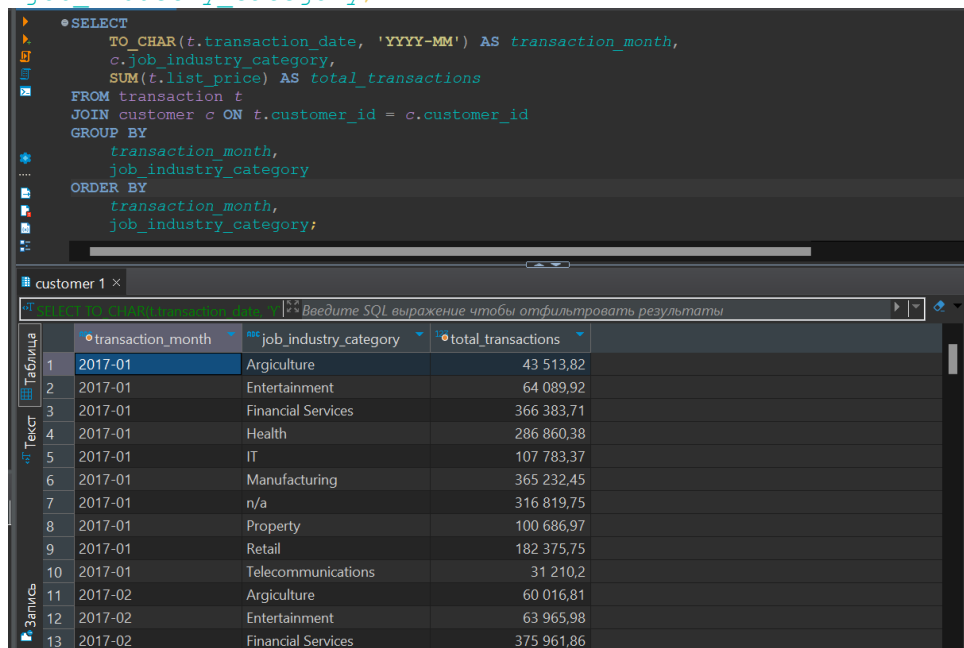
```
SELECT COUNT(customer_id), job_industry_category
FROM customer
GROUP BY job_industry_category
ORDER BY COUNT(customer_id) DESC;
```



	count	job_industry_category
1	799	Manufacturing
2	774	Financial Services
3	656	n/a
4	602	Health
5	358	Retail
6	267	Property
7	223	IT
8	136	Entertainment
9	113	Agriculture
10	72	Telecommunications

- Найти сумму транзакций за каждый месяц по сферам деятельности, отсортировав по месяцам и по сфере деятельности.

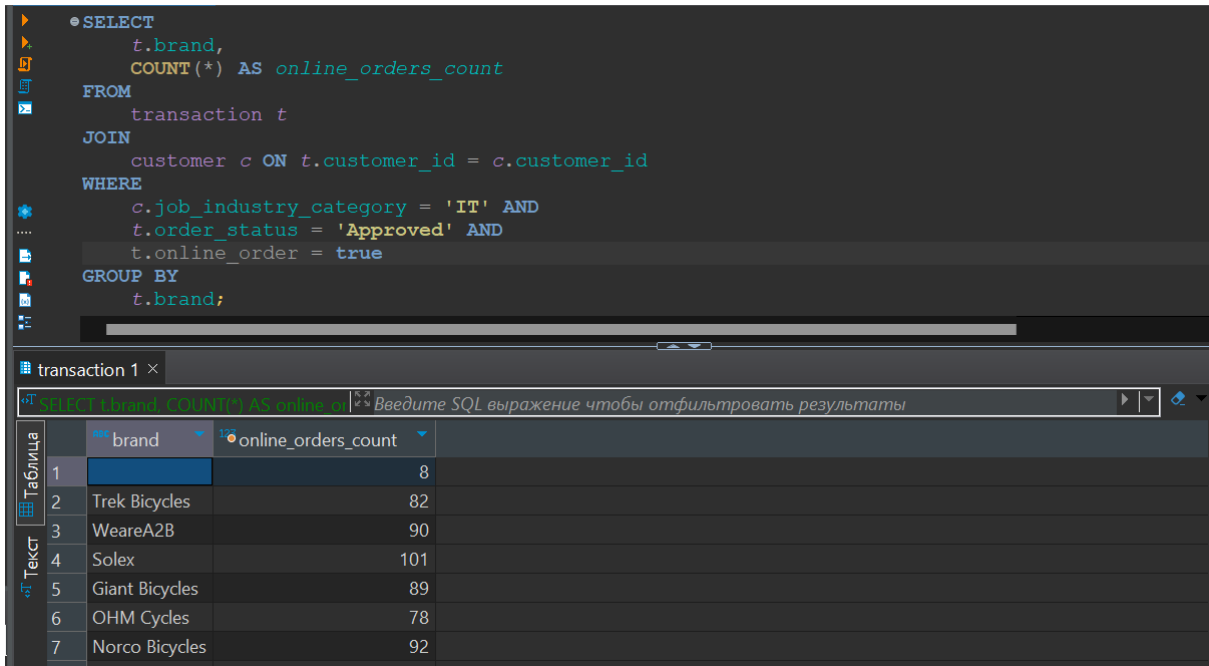
```
SELECT
    TO_CHAR(t.transaction_date, 'YYYY-MM') AS transaction_month,
    c.job_industry_category,
    SUM(t.list_price) AS total_transactions
FROM transaction t
JOIN customer c ON t.customer_id = c.customer_id
GROUP BY
    transaction_month,
    job_industry_category
ORDER BY
    transaction_month,
    job_industry_category;
```



	transaction_month	job_industry_category	total_transactions
1	2017-01	Agriculture	43 513.82
2	2017-01	Entertainment	64 089.92
3	2017-01	Financial Services	366 383.71
4	2017-01	Health	286 860.38
5	2017-01	IT	107 783.37
6	2017-01	Manufacturing	365 232.45
7	2017-01	n/a	316 819.75
8	2017-01	Property	100 686.97
9	2017-01	Retail	182 375.75
10	2017-01	Telecommunications	31 210.2
11	2017-02	Agriculture	60 016.81
12	2017-02	Entertainment	63 965.98
13	2017-02	Financial Services	375 961.86

- Вывести количество онлайн-заказов для всех брендов в рамках подтвержденных заказов клиентов из сферы IT.

```
SELECT
    t.brand,
    COUNT(*) AS online_orders_count
FROM
    transaction t
JOIN
    customer c ON t.customer_id = c.customer_id
WHERE
    c.job_industry_category = 'IT' AND
    t.order_status = 'Approved' AND
    t.online_order = true
GROUP BY
    t.brand;
```



The screenshot shows a SQL IDE interface. The top pane displays the SQL query. The bottom pane shows the results in a table view. The table has two columns: 'brand' and 'online_orders_count'. The results are as follows:

brand	online_orders_count
1	8
2	82
3	90
4	101
5	89
6	78
7	92

- Найти по всем клиентам сумму всех транзакций (list_price), максимум, минимум и количество транзакций, отсортировав результат по убыванию суммы транзакций и количества клиентов. Выполните двумя способами: используя только group by и используя только оконные функции. Сравните результат.

Первый способ (GROUP BY)

```
SELECT
    customer_id,
    SUM(list_price) AS total_transactions,
    MAX(list_price) AS max_transaction,
    MIN(list_price) AS min_transaction,
    COUNT(transaction_id) AS transaction_count
FROM
    transaction
GROUP BY
    customer_id
ORDER BY
    total_transactions DESC,
    transaction_count DESC;
```

•SELECT

```

customer_id,
SUM(list_price) AS total_transactions,
MAX(list_price) AS max_transaction,
MIN(list_price) AS min_transaction,
COUNT(transaction_id) AS transaction_count
FROM
transaction
GROUP BY
customer_id
ORDER BY
total_transactions DESC,
transaction_count DESC;

```

transaction 1 ×

SELECT customer_id, SUM(list_price) AS total_transactions, MAX(list_price) AS max_transaction, MIN(list_price) AS min_transaction, COUNT(transaction_id) AS transaction_count

	customer_id	total_transactions	max_transaction	min_transaction	transaction_count
4	1 597	18 052,68	2 091,47	360,4	12
5	941	17 898,46	2 091,47	1 057,51	10
6	2 788	17 258,94	2 083,94	183,86	11
7	936	17 160,24	2 005,66	183,86	12
8	1 887	17 133,93	2 091,47	688,63	11
9	1 302	17 035,83	1 977,36	71,16	13
10	1 140	16 199,24	2 083,94	183,86	13
11	2 309	16 122,34	2 091,47	290,62	12
12	729	15 826	2 091,47	586,45	10
13	1 103	15 447,92	1 977,36	230,91	12
14	1 317	15 370,81	2 091,47	569,56	9
15	2 871	15 001,01	2 005,66	544,05	11

Второй способ (Оконные функции)

```

SELECT
customer_id,
SUM(list_price) OVER (PARTITION BY customer_id) AS total_transactions,
MAX(list_price) OVER (PARTITION BY customer_id) AS max_transaction,
MIN(list_price) OVER (PARTITION BY customer_id) AS min_transaction,
COUNT(transaction_id) OVER (PARTITION BY customer_id) AS
transaction_count
FROM
transaction
ORDER BY
total_transactions DESC,
transaction_count DESC;

```

•SELECT

```

customer_id,
SUM(list_price) OVER (PARTITION BY customer_id) AS total_transactions,
MAX(list_price) OVER (PARTITION BY customer_id) AS max_transaction,
MIN(list_price) OVER (PARTITION BY customer_id) AS min_transaction,
COUNT(transaction_id) OVER (PARTITION BY customer_id) AS transaction_count
FROM
transaction
ORDER BY
total_transactions DESC,
transaction_count DESC;

```

transaction 1 ×

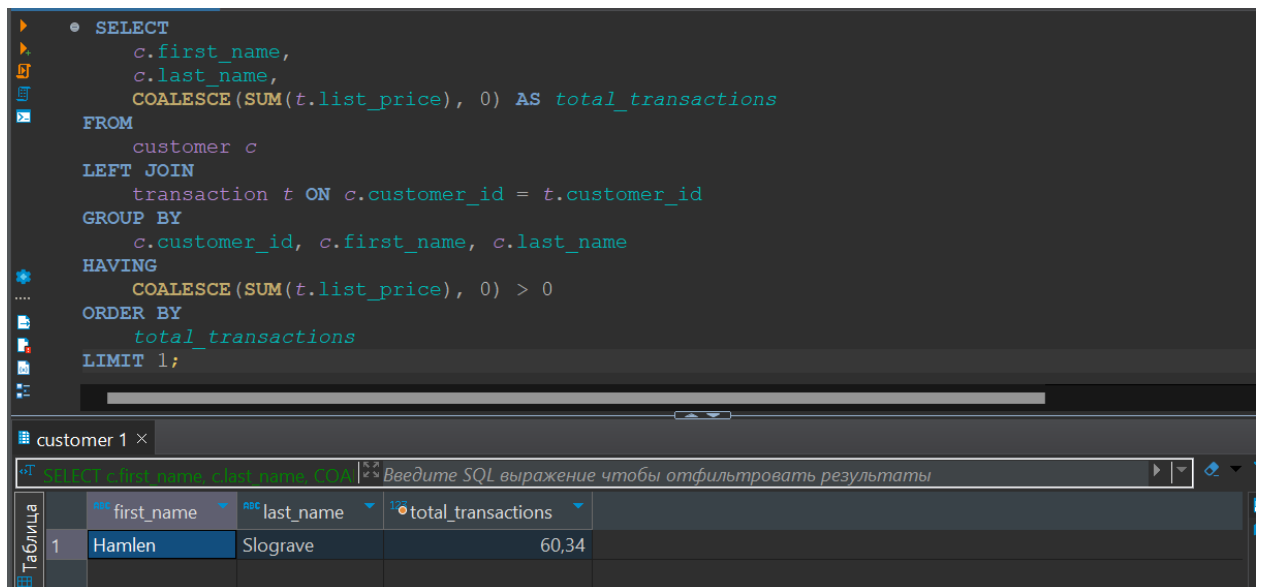
SELECT customer_id, SUM(list_price) AS total_transactions, MAX(list_price) AS max_transaction, MIN(list_price) AS min_transaction, COUNT(transaction_id) AS transaction_count

	customer_id	total_transactions	max_transaction	min_transaction	transaction_count
4	2 183	19 071,32	2 005,66	230,91	14
5	2 183	19 071,32	2 005,66	230,91	14
6	2 183	19 071,32	2 005,66	230,91	14
7	2 183	19 071,32	2 005,66	230,91	14
8	2 183	19 071,32	2 005,66	230,91	14
9	2 183	19 071,32	2 005,66	230,91	14
10	2 183	19 071,32	2 005,66	230,91	14
11	2 183	19 071,32	2 005,66	230,91	14
12	2 183	19 071,32	2 005,66	230,91	14
13	2 183	19 071,32	2 005,66	230,91	14
14	2 183	19 071,32	2 005,66	230,91	14
15	2 183	19 071,32	2 005,66	230,91	14
16	2 183	19 071,32	2 005,66	230,91	14
17	2 183	19 071,32	2 005,66	230,91	14

- Найти имена и фамилии клиентов с минимальной/максимальной суммой транзакций за весь период (сумма транзакций не может быть null). Напишите отдельные запросы для минимальной и максимальной суммы.

Минимум:

```
SELECT
    c.first_name,
    c.last_name,
    COALESCE(SUM(t.list_price), 0) AS total_transactions
FROM
    customer c
LEFT JOIN
    transaction t ON c.customer_id = t.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name
HAVING
    COALESCE(SUM(t.list_price), 0) > 0
ORDER BY
    total_transactions
LIMIT 1;
```



The screenshot shows a SQL IDE with a query editor and a results pane. The query in the editor is identical to the one above. The results pane shows a single row with the following data:

first_name	last_name	total_transactions
Hamlen	Slograve	60,34

Максимум:

```
SELECT
    c.first_name,
    c.last_name,
    COALESCE(SUM(t.list_price), 0) AS total_transactions
FROM
    customer c
LEFT JOIN
    transaction t ON c.customer_id = t.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name
ORDER BY
    total_transactions DESC
LIMIT 1;
```

```

SELECT
    c.first_name,
    c.last_name,
    COALESCE(SUM(t.list_price), 0) AS total_transactions
FROM
    customer c
LEFT JOIN
    transaction t ON c.customer_id = t.customer_id
GROUP BY
    c.customer_id, c.first_name, c.last_name
ORDER BY
    total_transactions DESC
LIMIT 1;

```

customer 1 ×

Введите SQL выражение чтобы отфильтровать результаты

	first_name	last_name	total_transactions
1	Jillie	Fyndon	19 071,32

- Вывести только самые первые транзакции клиентов. Решить с помощью оконных функций.

```

SELECT
    customer_id,
    transaction_id,
    transaction_date
FROM (
    SELECT
        customer_id,
        transaction_id,
        transaction_date,
        ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY
transaction_date) AS row_num
    FROM
        transaction
)
WHERE row_num = 1;

```

```

SELECT
    customer_id,
    transaction_id,
    transaction_date
FROM (
    SELECT
        customer_id,
        transaction_id,
        transaction_date,
        ROW_NUMBER() OVER (PARTITION BY customer_id ORDER BY transaction_date) AS row_num
    FROM
        transaction
)
WHERE

```

transaction 1 ×

Введите SQL выражение чтобы отфильтровать результаты

	customer_id	transaction_id	transaction_date
1	1	9 785	2017-01-05
2	2	2 261	2017-05-04
3	3	10 302	2017-02-23
4	4	12 441	2017-04-03
5	5	2 291	2017-03-03
6	6	7 096	2017-01-28
7	7	18 369	2017-02-18
8	8	10 792	2017-01-04
9	9	8 591	2017-02-04
10	10	5 956	2017-06-20
11	11	6 004	2017-04-02
12	12	6 741	2017-02-12

- Вывести имена, фамилии и профессии клиентов, между транзакциями которых был максимальный интервал (интервал вычисляется в днях)

```
WITH TransactionIntervals AS (  
    SELECT  
        customer_id,  
        transaction_date,  
        LAG(transaction_date) OVER (PARTITION BY customer_id ORDER BY  
transaction_date) AS prev_transaction_date,  
        COALESCE(transaction_date - LAG(transaction_date) OVER (PARTITION  
BY customer_id ORDER BY transaction_date), 0) AS interval_days  
    FROM  
        transaction  
)  
  
SELECT  
    c.first_name,  
    c.last_name,  
    c.job_title,  
    ti.interval_days  
FROM  
    customer c  
JOIN  
    TransactionIntervals ti ON c.customer_id = ti.customer_id  
ORDER BY  
    ti.interval_days DESC  
LIMIT 1;
```

The screenshot shows a SQL IDE interface. The top pane displays the SQL query, which is identical to the one in the previous block. The bottom pane shows the results of the query in a table format. The table has four columns: first_name, last_name, job_title, and interval_days. The first row of data shows 'Susanetta' as the first name, an empty last name, 'Legal Assistant' as the job title, and '357' as the interval in days. The interface also includes a sidebar on the left with icons for various database operations and a bottom status bar with a search prompt in Russian.

	first_name	last_name	job_title	interval_days
1	Susanetta		Legal Assistant	357