

## Вариант №1

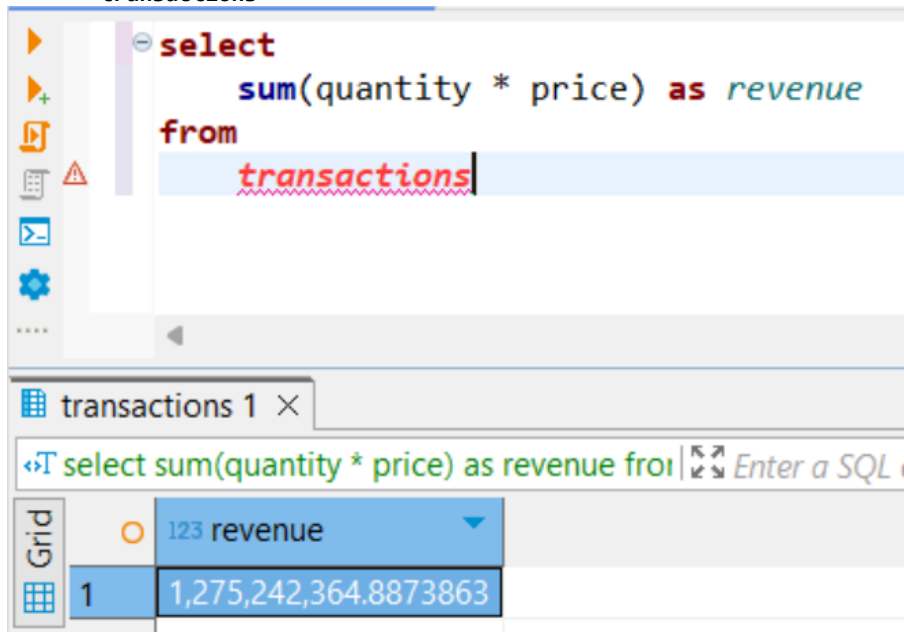
<https://www.youtube.com/shorts/9xzX4ACDsEI>

- Создание таблицы transaction и ее наполнение

```
create table transactions (  
    transaction_id uint32,  
    user_id uint32,  
    product_id uint32,  
    quantity uint8,  
    price float32,  
    transaction_date date  
) engine = mergetree()  
order by (transaction_id);  
  
insert into transactions  
select  
    number as transaction_id,  
    (intdiv(rand(),  
429496) % 10000) + 1 as user_id,  
    (intdiv(rand(),  
429496) % 1000) + 1 as product_id,  
    (intdiv(rand(),  
429496) % 100) + 1 as quantity,  
    round(((rand() % 49500) / 100) + 5,  
2) as price,  
    todate('2023-01-01') + (intdiv(rand(),  
429496) % 365) as transaction_date  
from  
    numbers(100000)
```

- Рассчитайте общий доход от всех операций

```
select  
    sum(quantity * price) as revenue  
from  
    transactions
```



- Найдите средний доход с одной сделки

```
select round(avg(revenue), 2) as average_revenue
from (
    select transaction_id, sum(quantity * price) as revenue
    from transactions
    group by transaction_id
)
```

The screenshot shows a SQL IDE with a query editor and a results grid. The query is: `SELECT ROUND(AVG(revenue), 2) AS average_revenue FROM (SELECT transaction_id, SUM(quantity * price) AS revenue FROM transactions GROUP BY transaction_id)`. The results grid shows one row with the value 12,752.42.

Grid	123 average_revenue
1	12,752.42

- Определите общее количество проданной продукции

```
select
    sum(quantity) as total_sold
from
    transactions
```

The screenshot shows a SQL IDE with a query editor and a results grid. The query is: `SELECT SUM(quantity) AS total_sold FROM transactions`. The results grid shows one row with the value 5,042,331.

Grid	123 total_sold
1	5,042,331

- Подсчитайте количество уникальных пользователей, совершивших покупку

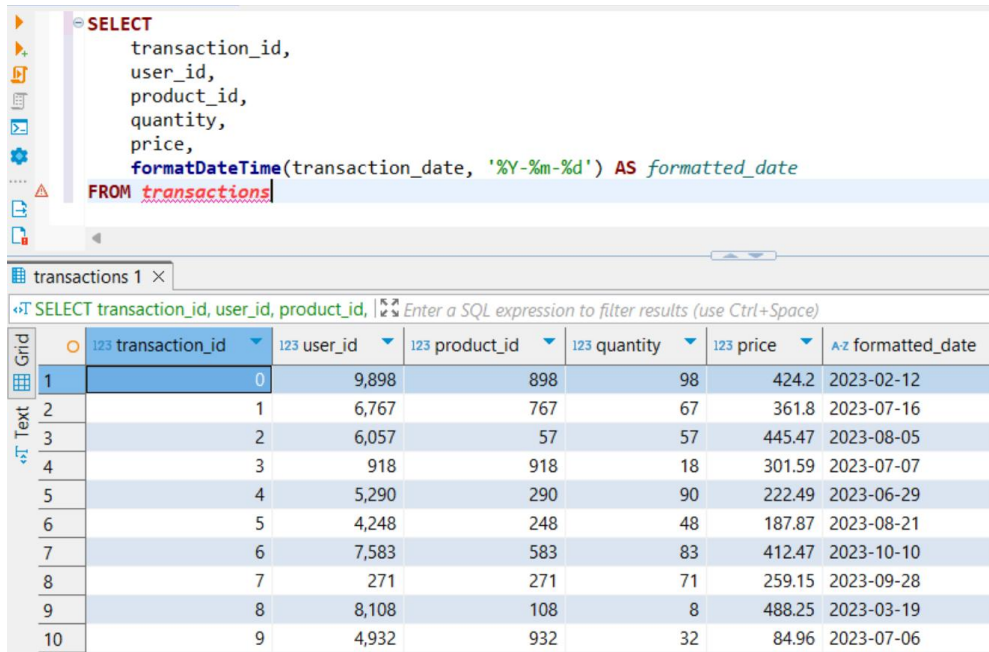
```
select
    uniqexact(user_id) as unique_users
from
    transactions
```

The screenshot shows a SQL IDE with a query editor and a results grid. The query is: `SELECT uniqExact(user_id) AS unique_users FROM transactions`. The results grid shows one row with the value 9,999.

Grid	123 unique_users
1	9,999

- Преобразуйте `transaction\_date` в строку формата `YYYY-MM-DD`

```
select
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    formatdatetime(transaction_date, '%Y-%m-%d') as formatted_date
from transactions
```



The screenshot shows a SQL query window with the following code:

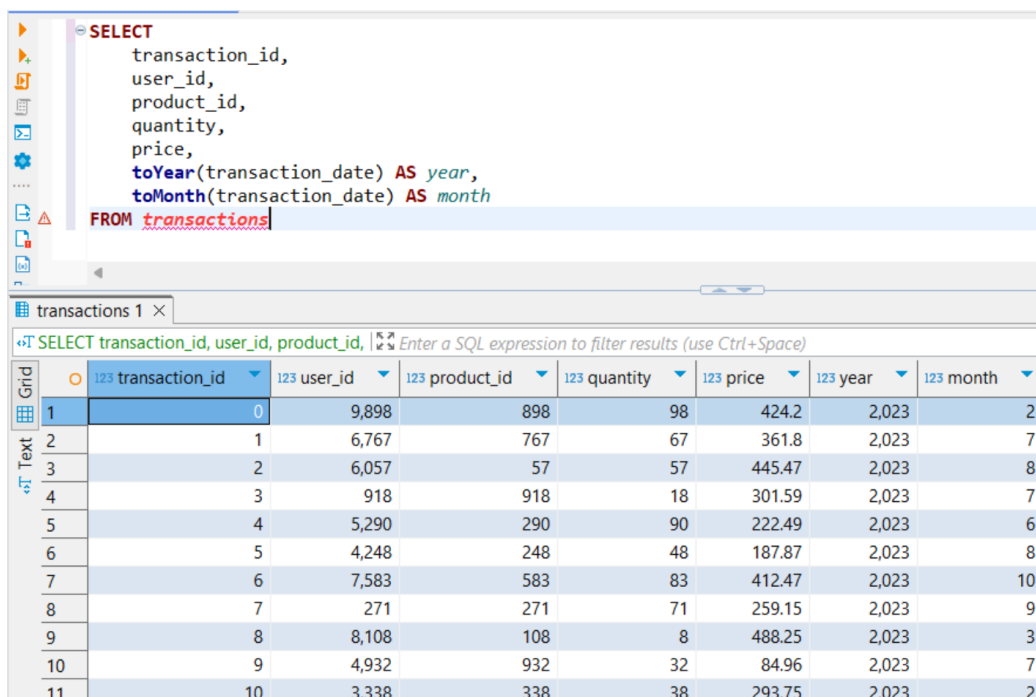
```
SELECT
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    formatDateTime(transaction_date, '%Y-%m-%d') AS formatted_date
FROM transactions
```

Below the query window is a results grid titled "transactions 1 x". The grid contains 10 rows of data. The columns are: transaction\_id, user\_id, product\_id, quantity, price, and formatted\_date.

	transaction_id	user_id	product_id	quantity	price	formatted_date
1	0	9,898	898	98	424.2	2023-02-12
2	1	6,767	767	67	361.8	2023-07-16
3	2	6,057	57	57	445.47	2023-08-05
4	3	918	918	18	301.59	2023-07-07
5	4	5,290	290	90	222.49	2023-06-29
6	5	4,248	248	48	187.87	2023-08-21
7	6	7,583	583	83	412.47	2023-10-10
8	7	271	271	71	259.15	2023-09-28
9	8	8,108	108	8	488.25	2023-03-19
10	9	4,932	932	32	84.96	2023-07-06

- Извлеките год и месяц из `transaction\_date`

```
select
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    toyear(transaction_date) as year,
    tomonth(transaction_date) as month
from transactions
```



The screenshot shows a SQL query window with the following code:

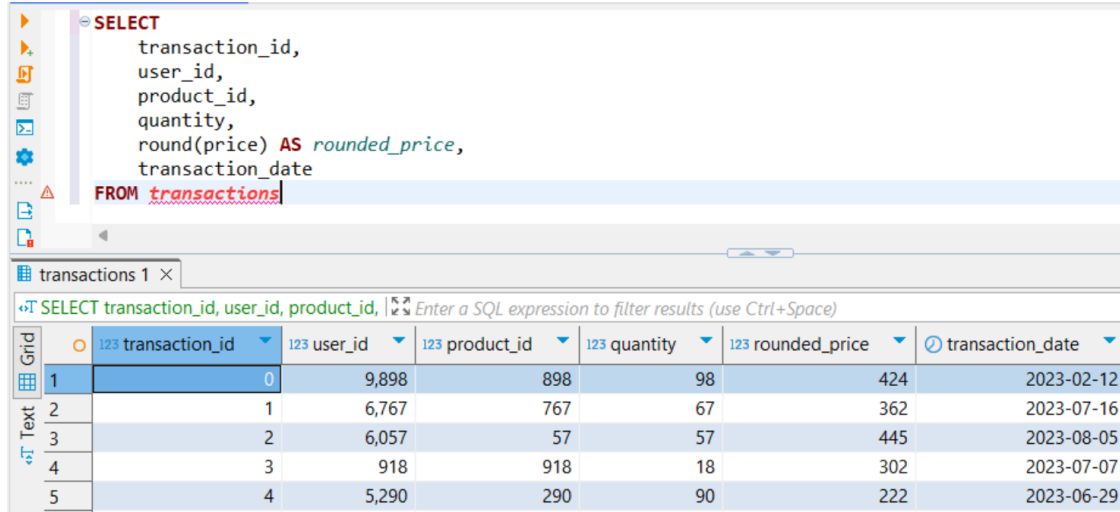
```
SELECT
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    toYear(transaction_date) AS year,
    toMonth(transaction_date) AS month
FROM transactions
```

Below the query window is a results grid titled "transactions 1 x". The grid contains 11 rows of data. The columns are: transaction\_id, user\_id, product\_id, quantity, price, year, and month.

	transaction_id	user_id	product_id	quantity	price	year	month
1	0	9,898	898	98	424.2	2,023	2
2	1	6,767	767	67	361.8	2,023	7
3	2	6,057	57	57	445.47	2,023	8
4	3	918	918	18	301.59	2,023	7
5	4	5,290	290	90	222.49	2,023	6
6	5	4,248	248	48	187.87	2,023	8
7	6	7,583	583	83	412.47	2,023	10
8	7	271	271	71	259.15	2,023	9
9	8	8,108	108	8	488.25	2,023	3
10	9	4,932	932	32	84.96	2,023	7
11	10	3,338	338	38	293.75	2,023	2

- Округлите `price` до ближайшего целого числа

```
select
    transaction_id,
    user_id,
    product_id,
    quantity,
    round(price) as rounded_price,
    transaction_date
from transactions
```



The screenshot shows a SQL IDE with the following query:

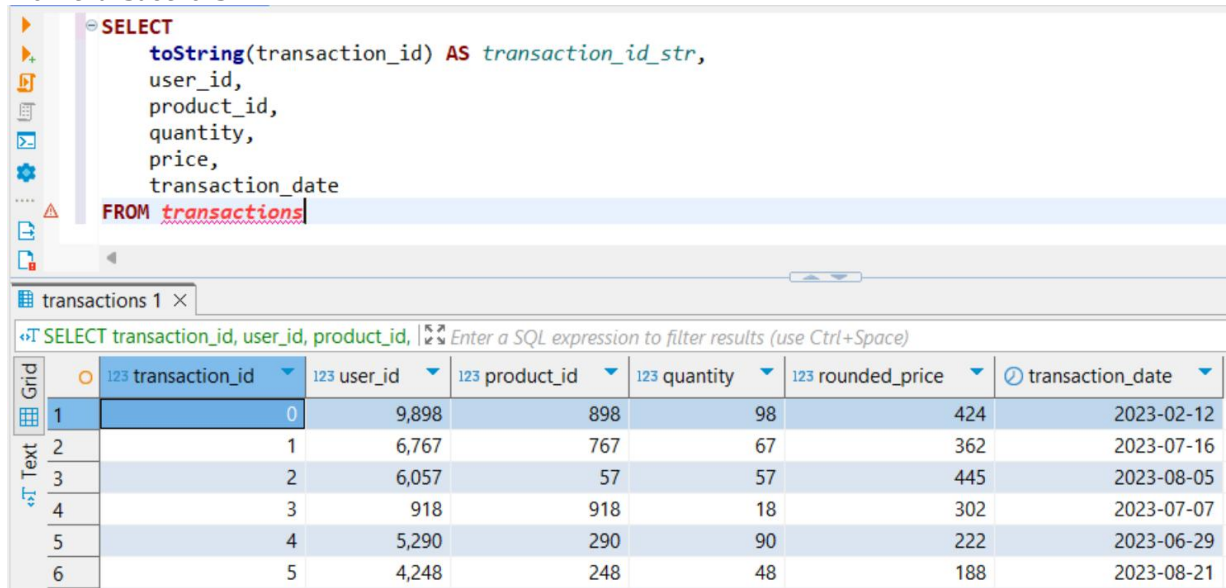
```
SELECT
    transaction_id,
    user_id,
    product_id,
    quantity,
    round(price) AS rounded_price,
    transaction_date
FROM transactions
```

The results are displayed in a table with 7 columns: transaction\_id, user\_id, product\_id, quantity, rounded\_price, and transaction\_date. The first 5 rows of data are shown.

	transaction_id	user_id	product_id	quantity	rounded_price	transaction_date
1	0	9,898	898	98	424	2023-02-12
2	1	6,767	767	67	362	2023-07-16
3	2	6,057	57	57	445	2023-08-05
4	3	918	918	18	302	2023-07-07
5	4	5,290	290	90	222	2023-06-29

- Преобразуйте `transaction\_id` в строку

```
select
    toString(transaction_id) as transaction_id_str,
    user_id,
    product_id,
    quantity,
    price,
    transaction_date
from transactions
```



The screenshot shows a SQL IDE with the following query:

```
SELECT
    toString(transaction_id) AS transaction_id_str,
    user_id,
    product_id,
    quantity,
    price,
    transaction_date
FROM transactions
```

The results are displayed in a table with 7 columns: transaction\_id\_str, user\_id, product\_id, quantity, rounded\_price, and transaction\_date. The first 6 rows of data are shown.

	transaction_id_str	user_id	product_id	quantity	rounded_price	transaction_date
1	0	9,898	898	98	424	2023-02-12
2	1	6,767	767	67	362	2023-07-16
3	2	6,057	57	57	445	2023-08-05
4	3	918	918	18	302	2023-07-07
5	4	5,290	290	90	222	2023-06-29
6	5	4,248	248	48	188	2023-08-21

- Создайте простую UDF для расчета общей стоимости транзакции

```
create function calculate_total_cost as (quantity, price) ->
quantity * price;
```

<pre>CREATE FUNCTION calculate_total_cost AS (quantity, price) -&gt; quantity * price;</pre>	
Statistics 1	
Name	Value
Updated Rows	0
Query	CREATE FUNCTION calculate_total_cost AS (quantity, price) -> quantity * price
Start time	Mon Oct 21 13:40:09 MSK 2024
Finish time	Mon Oct 21 13:40:09 MSK 2024

- Используйте созданную UDF для расчета общей цены для каждой транзакции

```
select
transaction_id,
user_id,
product_id,
quantity,
price,
calculate_total_cost(quantity, price) as total_cost
from transactions;
```

```
CREATE FUNCTION calculate_total_cost AS (quantity, price) ->
    quantity * price;

SELECT
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    calculate_total_cost(quantity, price) AS total_cost
FROM transactions;
```

transactions 1

SELECT transaction\_id, user\_id, product\_id, quantity, price, total\_cost

	transaction_id	user_id	product_id	quantity	price	total_cost
1	0	9,898	898	98	424.2	41,571.6011962891
2	1	6,767	767	67	361.8	24,240.5991821289
3	2	6,057	57	57	445.47	25,391.7900695801
4	3	918	918	18	301.59	5,428.619934082
5	4	5,290	290	90	222.49	20,024.1004943848

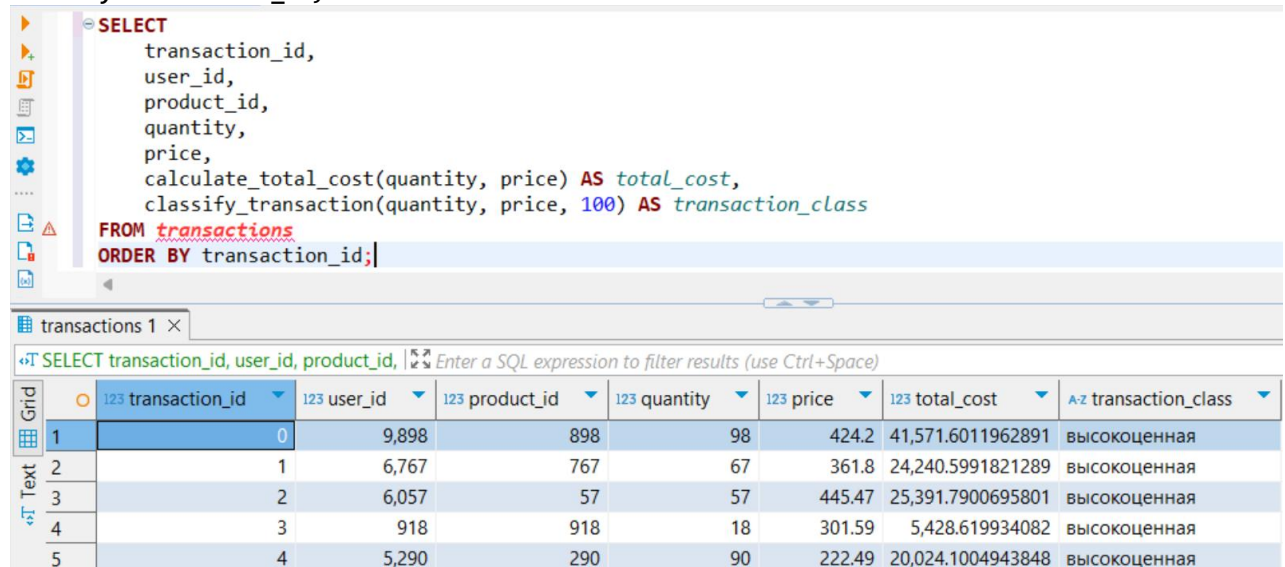
- Создайте UDF для классификации транзакций на «высокоценные» и «малоценные» на основе порогового значения (например, 100)

```
create function classify_transaction as (quantity, price, threshold) ->
if(quantity * price >= threshold, 'высокоценная', 'малоценная');
```

<pre>CREATE FUNCTION classify_transaction AS (quantity, price, threshold) -&gt; if(quantity * price &gt;= threshold, 'высокоценная', 'малоценная');</pre>	
Statistics 1	
Name	Value
Updated Rows	0
Query	CREATE FUNCTION classify_transaction AS (quantity, price, threshold) -> if(quantity * price >= threshold, 'высокоценная', 'малоценная')
Start time	Mon Oct 21 13:45:42 MSK 2024
Finish time	Mon Oct 21 13:45:42 MSK 2024

- Примените UDF для категоризации каждой транзакции

```
select
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    calculate_total_cost(quantity, price) as total_cost,
    classify_transaction(quantity, price, 100) as transaction_class
from transactions
order by transaction_id;
```



The screenshot shows a database IDE interface. The top pane displays a SQL query that selects transaction details and calculates a total cost and a transaction class. The bottom pane shows the results of this query in a grid view with 8 columns: transaction\_id, user\_id, product\_id, quantity, price, total\_cost, and transaction\_class. The results are ordered by transaction\_id.

	transaction_id	user_id	product_id	quantity	price	total_cost	transaction_class
1	0	9,898	898	98	424.2	41,571.6011962891	высокоценная
2	1	6,767	767	67	361.8	24,240.5991821289	высокоценная
3	2	6,057	57	57	445.47	25,391.7900695801	высокоценная
4	3	918	918	18	301.59	5,428.619934082	высокоценная
5	4	5,290	290	90	222.49	20,024.1004943848	высокоценная