1. **Агрегатные функции**
    1. Рассчитайте общий доход от всех операций.
    2. Найдите средний доход с одной сделки.
    3. Определите общее количество проданной продукции.
    4. Подсчитайте количество уникальных пользователей, совершивших покупку.
2. **Функции для работы с типами данных**
    1. Преобразуйте `transaction_date` в **строку** формата `YYYY-MM-DD`.
    2. Извлеките год и месяц из `transaction_date`.
    3. Округлите `price` до ближайшего целого числа.
    4. Преобразуйте `transaction_id` в строку.
3. **User-Defined Functions (UDFs)**
    1. Создайте простую UDF для расчета общей стоимости транзакции.
    2. Используйте созданную UDF для расчета общей цены для каждой транзакции.
    3. Создайте UDF для классификации транзакций на «высокоценные» и «малоценные» на основе порогового значения (например, 100).
    4. Примените UDF для категоризации каждой транзакции.

```
ch1 :) insert into transactions SELECT
transaction_id,
user_id,
product_id,
quantity,
price,
toDate(transaction_date) AS transaction_date
FROM file("/var/lib/clickhouse/user_files/DataSet_1.csv")

INSERT INTO transactions SELECT
    transaction_id,
    user_id,
    product_id,
    quantity,
    price,
    toDate(transaction_date) AS transaction_date
FROM file(`/var/lib/clickhouse/user_files/DataSet_1.csv`)

Query id: df848156-859c-408e-98b3-90f01fc8615d

Ok.

0 rows in set. Elapsed: 0.014 sec. Processed 1.13 thousand rows, 51.93 KB (80.76 thousand rows/s., 3.70 MB/s.)
Peak memory usage: 197.90 KiB.

ch1 :) select * from transactions

SELECT *
FROM transactions

Query id: c34a573d-4bc2-4fb0-a1a2-9d762cd08d74

    ┌─transaction_id─┬─user_id─┬─product_id─┬─quantity─┬─price─┬─transaction_date─┐
 1. │              1 │     108 │       1007 │       69 │ 1269.9 │       2023-05-02 │
 2. │              2 │      28 │       1015 │       93 │  775.4 │       2023-03-16 │
 3. │              3 │     347 │       1013 │        1 │  815.9 │       2023-03-17 │
 4. │              4 │     700 │       1030 │       52 │  773.4 │       2023-03-18 │
 5. │              5 │     205 │       1011 │      176 │    929 │       2023-03-19 │
 6. │              6 │     747 │       1006 │       72 │ 1954.7 │       2023-03-20 │
 7. │              7 │     381 │       1008 │      181 │ 1826.1 │       2023-03-21 │
 8. │              8 │     596 │       1012 │       90 │  480.1 │       2023-03-22 │
 9. │              9 │     957 │       1002 │      194 │ 1675.8 │       2023-03-23 │
10. │             10 │     823 │       1020 │       71 │  506.5 │       2023-03-24 │
11. │             11 │     210 │       1002 │       47 │   1370 │       2023-03-25 │
12. │             12 │     253 │       1014 │      119 │  731.8 │       2023-03-26 │
```

Рассчитайте общий доход от всех операций.

```
ch1 :) select sum(quantity*price) from transactions

SELECT sum(quantity * price)
FROM transactions

Query id: c0c739ef-bfbd-4a7c-a3ee-f739ed90dc90

    ┌─sum(multiply(quantity, price))─┐
1.  │           129151625.71524048  │ -- 129.15 million
    └────────────────────────────────┘

1 row in set. Elapsed: 0.020 sec. Processed 2.27 thousand rows, 6.80 KB (112.89 thousand rows/s., 338.68 KB/s.)
Peak memory usage: 65.17 KiB.

ch1 :)
```

Найдите средний доход с одной сделки.

```
ch1 :) select avg(quantity*price) from transactions

SELECT avg(quantity * price)
FROM transactions

Query id: 540fdbdd-331e-47f4-bbc8-8e901f276cbf

    ┌─avg(multiply(quantity, price))─┐
1.  │           113990.84352624931  │
    └────────────────────────────────┘

1 row in set. Elapsed: 0.022 sec. Processed 2.27 thousand rows, 6.80 KB (104.49 thousand rows/s., 313.46 KB/s.)
Peak memory usage: 65.03 KiB.

ch1 :)
```

Определите общее количество проданной продукции.

```
ch1 :) select sum(quantity) from transactions

SELECT sum(quantity)
FROM transactions

Query id: 19dc6f0b-9835-44b3-a029-af47e62f2a54

    ┌─sum(quantity)─┐
1.  │        113749 │
    └───────────────┘

1 row in set. Elapsed: 0.009 sec. Processed 2.27 thousand rows, 2.27 KB (265.94
Peak memory usage: 64.78 KiB.

ch1 :)
```

Подсчитайте количество уникальных пользователей, совершивших покупку.

```
ch1 :) select count(DISTINCT user_id) from transactions

SELECT countDistinct(user_id)
FROM transactions

Query id: d599da91-2ada-47dc-bda2-a31d246fceb5

    ┌─countDistinct(user_id)─┐
1.  │                    691 │
    └────────────────────────┘

1 row in set. Elapsed: 0.054 sec. Processed 2.27 thousand rows, 5.67 KB (42.11 thousand rows/s., 105.28 KB/s.)
Peak memory usage: 65.31 KiB.

ch1 :)
```

Преобразуйте `transaction_date` в **строку** формата `YYYY-MM-DD`.

```
ch1 :) select toString(toDate(transaction_date)) from transactions

SELECT toString(toDate(transaction_date))
FROM transactions

Query id: e8573508-5d59-432a-9e01-94df9c9621fd

        ┌─toString(toDate(transaction_date))─┐
   1.   │ 2023-05-02                         │
   2.   │ 2023-03-16                         │
   3.   │ 2023-03-17                         │
   4.   │ 2023-03-18                         │
   5.   │ 2023-03-19                         │
   6.   │ 2023-03-20                         │
   7.   │ 2023-03-21                         │
   8.   │ 2023-03-22                         │
   9.   │ 2023-03-23                         │
  10.   │ 2023-03-24                         │
  11.   │ 2023-03-25                         │
  12.   │ 2023-03-26                         │
  13.   │ 2023-03-27                         │
  14.   │ 2023-03-28                         │
  15.   │ 2023-03-29                         │
  16.   │ 2023-03-30                         │
  17.   │ 2023-03-31                         │
  18.   │ 2023-04-01                         │
  19.   │ 2023-04-02                         │
```

Извлеките год и месяц из `transaction_date`.

```
ch1 :) select toYear(transaction_date) as Year,toMonth(transaction_date) as Month from transactions

SELECT
    toYear(transaction_date) AS Year,
    toMonth(transaction_date) AS Month
FROM transactions

Query id: 13788268-8aa0-495b-a43a-e1d839e5cafd

        ┌─Year─┬─Month─┐
   1.   │ 2023 │     5 │
   2.   │ 2023 │     3 │
   3.   │ 2023 │     3 │
   4.   │ 2023 │     3 │
   5.   │ 2023 │     3 │
   6.   │ 2023 │     3 │
   7.   │ 2023 │     3 │
   8.   │ 2023 │     3 │
   9.   │ 2023 │     3 │
  10.   │ 2023 │     3 │
  11.   │ 2023 │     3 │
  12.   │ 2023 │     3 │
```

Округлите `price` до ближайшего целого числа.

```
ch1 :) select round(price) from transactions

SELECT round(price)
FROM transactions

Query id: dd9fd040-d0d3-4687-99c0-65ef828c0308

    ┌─round(price)─┐
 1. │         1270 │
 2. │          775 │
 3. │          816 │
 4. │          773 │
 5. │          929 │
 6. │         1955 │
 7. │         1826 │
 8. │          480 │
 9. │         1676 │
10. │          506 │
11. │         1370 │
12. │          732 │
13. │         1765 │
14. │         1369 │
15. │          372 │
16. │         1989 │
```

Преобразуйте `transaction_id` в строку.

```
ch1 :) select toString(transaction_id) from transactions

SELECT toString(transaction_id)
FROM transactions

Query id: ce84757e-30fa-4559-8d34-8a62e033d654

    ┌─toString(transaction_id)─┐
 1. │ 1                        │
 2. │ 2                        │
 3. │ 3                        │
 4. │ 4                        │
 5. │ 5                        │
 6. │ 6                        │
 7. │ 7                        │
 8. │ 8                        │
 9. │ 9                        │
10. │ 10                       │
11. │ 11                       │
12. │ 12                       │
13. │ 13                       │
14. │ 14                       │
15. │ 15                       │
16. │ 16                       │
```

Создайте простую UDF для расчета общей стоимости транзакции.
Используйте созданную UDF для расчета общей цены для каждой транзакции.

```
ch1 :) CREATE FUNCTION sum_tran AS (a,b) -> a*b

CREATE FUNCTION sum_tran AS (a, b) -> (a * b)

Query id: 2b65f2bb-640f-4493-af31-f27cecd445f6

Ok.

0 rows in set. Elapsed: 0.023 sec.

ch1 :) select sum_tran(price,quanti)
Display all 511 possibilities? (y or n)
ch1 :) select sum_tran(price,quantity) from transactions

SELECT sum_tran(price, quantity)
FROM transactions

Query id: 67f1e0fa-542f-4459-bd62-9b123fb456ff

    ┌─sum_tran(price, quantity)─┐
 1. │          87623.10168457031 │
 2. │          72112.20227050781 │
 3. │          815.9000244140625 │
 4. │          40216.80126953125 │
 5. │                      163504 │
 6. │           140738.396484375 │
 7. │          330524.0955810547 │
 8. │         43209.000549316406 │
 9. │         325105.20947265625 │
10. │                    35961.5 │
11. │                      64390 │
12. │          87084.19854736328 │
```

Создайте UDF для классификации транзакций на «высокоценные» и «малоценные» на основе порогового значения (например, 100).
Примените UDF для категоризации каждой транзакции.

```
ch1 :) CREATE FUNCTION ocenka AS (a) -> if(a>1000,'много','мало')

CREATE FUNCTION ocenka AS a -> if(a > 1000, 'много', 'мало')

Query id: bbf263dc-541b-44db-8018-22d92b7a64f6

Ok.

0 rows in set. Elapsed: 0.009 sec.

ch1 :) select ocenka(price),price from transactions

SELECT
    ocenka(price),
    price
FROM transactions

Query id: d874251e-e72e-4f3a-a0e0-9966cede2cc3

      ┌─ocenka(price)─┬─price─┐
   1. │ много         │ 1269.9 │
   2. │ мало          │  775.4 │
   3. │ мало          │  815.9 │
   4. │ мало          │  773.4 │
   5. │ мало          │    929 │
   6. │ много         │ 1954.7 │
   7. │ много         │ 1826.1 │
   8. │ мало          │  480.1 │
   9. │ много         │ 1675.8 │
  10. │ мало          │  506.5 │
  11. │ много         │   1370 │
  12. │ мало          │  731.8 │
  13. │ много         │   1765 │
  14. │ много         │ 1369.1 │
  15. │ мало          │  372.4 │
  16. │ много         │ 1988.9 │
  17. │ мало          │  897.1 │
  18. │ мало          │  763.7 │
  19. │ много         │ 1245.7 │
  20. │ много         │ 1670.6 │
  21. │ много         │   1483 │
```