**04 ДЗ - Функции для работы с типами данных, агрегатные функции и UDF.**

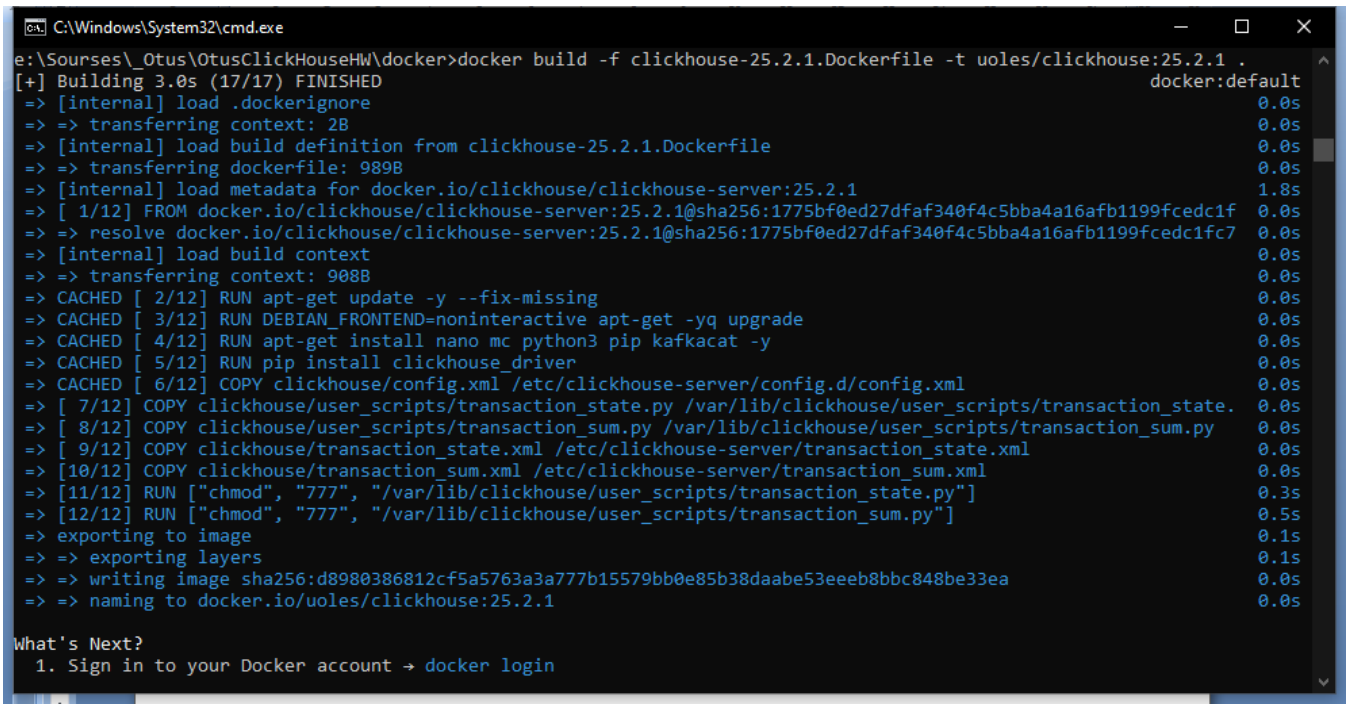
**Собираем Dockerfile для clickhouse.**

Dockerfile:

```
FROM clickhouse/clickhouse-server:25.2.1
MAINTAINER Maksim Kulikov <max.uoles@rambler.ru>

RUN apt-get update -y --fix-missing
RUN DEBIAN_FRONTEND=noninteractive apt-get -yq upgrade
RUN apt-get install nano mc python3 pip kafkacat -y
RUN pip install clickhouse_driver

COPY clickhouse/config.xml /etc/clickhouse-server/config.d/config.xml

COPY clickhouse/user_scripts/transaction_state.py /var/lib/clickhouse/user_scripts/transaction_state.py
COPY clickhouse/user_scripts/transaction_sum.py /var/lib/clickhouse/user_scripts/transaction_sum.py

COPY clickhouse/transaction_state.xml /etc/clickhouse-server/transaction_state.xml
COPY clickhouse/transaction_sum.xml /etc/clickhouse-server/transaction_sum.xml

RUN ["chmod", "+x", "/var/lib/clickhouse/user_scripts/transaction_state.py"]
RUN ["chmod", "+x", "/var/lib/clickhouse/user_scripts/transaction_sum.py"]

EXPOSE 8123 9000

ENTRYPOINT ["/entrypoint.sh"]
```

Переходим в папку docker и собираем образ командой:

docker build -f clickhouse-25.2.1.Dockerfile -t uoles/clickhouse:25.2.1 .



Запускаем контейнер командой:

```
docker run -d ^
  -p 18123:8123 ^
  -p 19000:9000 ^
  -e CLICKHOUSE_DB=my_database ^
  -e CLICKHOUSE_USER=username ^
  -e CLICKHOUSE_DEFAULT_ACCESS_MANAGEMENT=1 ^
  -e CLICKHOUSE_PASSWORD=password ^
  --name clickhouse-server ^
  --ulimit nofile=262144:262144 ^
uoles/clickhouse:25.2.1
```


**Создаем таблицу с тестовыми данными.**

Создаем таблицу:

```
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);
```

Заполняем тестовыми данными:

```
INSERT INTO transactions
	SELECT
		rand32() AS transaction_id,
		randUniform(1,1000)::Int AS user_id,
		randUniform(1,1000)::Int AS product_id,
		randUniform(1,10)::Int AS cnt,
		round( randUniform(15.5, 299.99), 2 ) AS price,
		now() - toIntervalSecond(rand() % (365 * 24 * 60 * 60)) AS datetime
	FROM system.numbers
	LIMIT 1000;
```

```
SELECT *
FROM transactions
```

transactions 1

SELECT * FROM transactions | Введите SQL выражение чтобы отфильтровать результаты

| | transaction_id | user_id | product_id | quantity | price | transaction_date |
|---|---|---|---|---|---|---|
| 182 | 773 120 441 | 43 | 43 | 3 | 214,71 | 2024-10-23 |
| 183 | 777 122 156 | 689 | 689 | 1 | 216,74 | 2024-09-06 |
| 184 | 778 376 991 | 864 | 864 | 4 | 264,61 | 2024-08-23 |
| 185 | 788 150 486 | 225 | 225 | 9 | 80,52 | 2024-05-02 |
| 186 | 788 906 878 | 294 | 294 | 4 | 137,4 | 2025-04-23 |
| 187 | 811 308 641 | 717 | 717 | 4 | 50,1 | 2024-08-07 |
| 188 | 811 757 571 | 652 | 652 | 9 | 128,16 | 2024-08-01 |
| 189 | 812 728 753 | 916 | 916 | 2 | 223,02 | 2024-07-21 |
| 190 | 817 091 298 | 908 | 908 | 6 | 291,65 | 2024-06-01 |

**Составляем запросы:**

- Рассчитайте общий доход от всех операций.

select sum(price * quantity)
from transactions

```
-- Рассчитайте общий доход от всех операций.
select sum(price * quantity)
from transactions
```

transactions 1

select sum(price * quantity) from transactions | Введите SQL выражение что

| | sum(multiply(price, quantity)) |
|---|---|
| 1 | 744 919,8599624634 |

- Найдите средний доход с одной сделки.

select avg(price * quantity)
from transactions

```
-- Найдите средний доход с одной сделки.
select avg(price * quantity)
from transactions
```

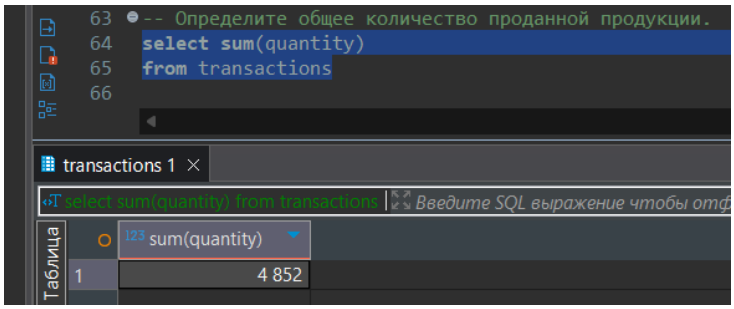
transactions 1

select avg(price * quantity) from transactions | Введите SQL выраже

| | avg(multiply(price, quantity)) |
|---|---|
| 1 | 744,9198599625 |

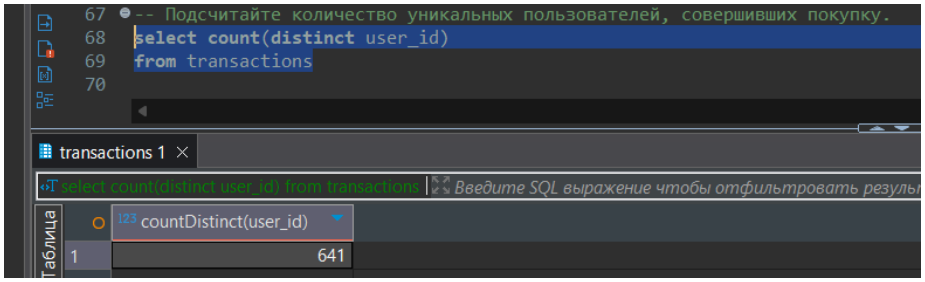
- Определите общее количество проданной продукции.

select sum(quantity)
from transactions



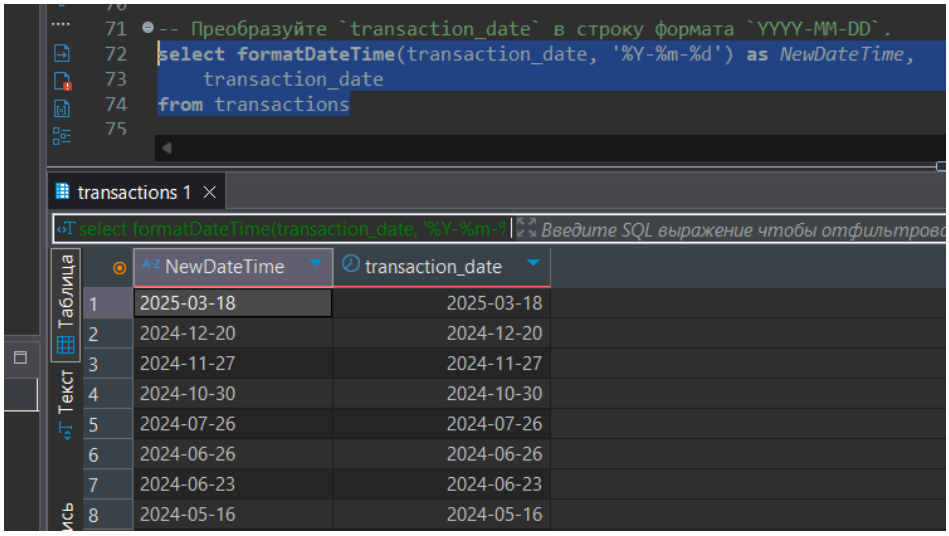

- Подсчитайте количество уникальных пользователей, совершивших покупку.

select count(distinct user_id)
from transactions



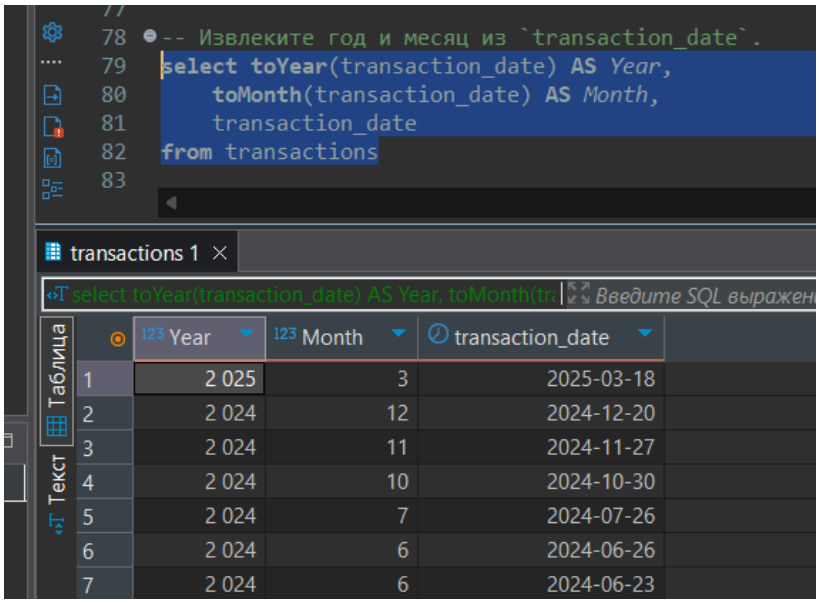

- Преобразуйте `transaction_date` в строку формата `YYYY-MM-DD`.

select formatDateTime(transaction_date, '%Y-%m-%d') as NewDateTime,
transaction_date
from transactions



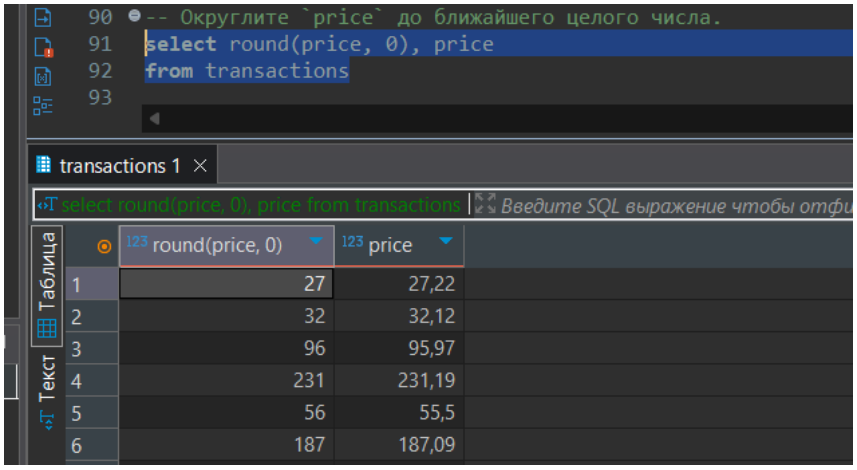
- Извлеките год и месяц из `transaction_date`.

```
select toYear(transaction_date) AS Year,
        toMonth(transaction_date) AS Month,
        transaction_date
from transactions
```



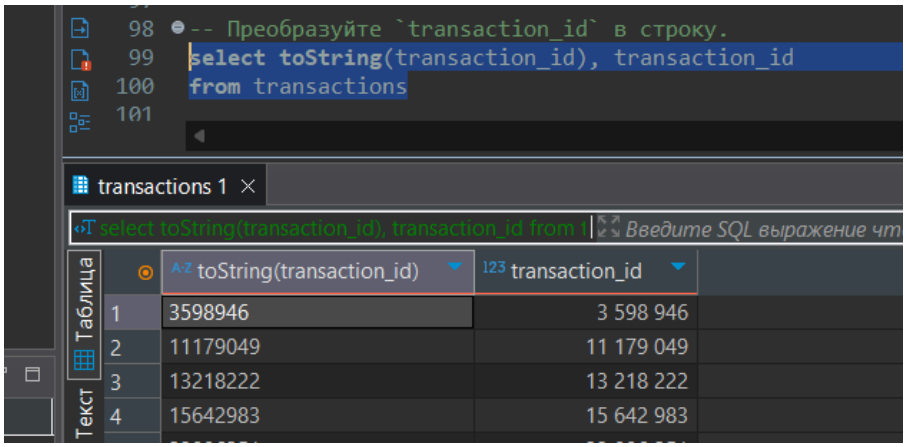

- Округлите `price` до ближайшего целого числа.

```
select round(price, 0), price
from transactions
```



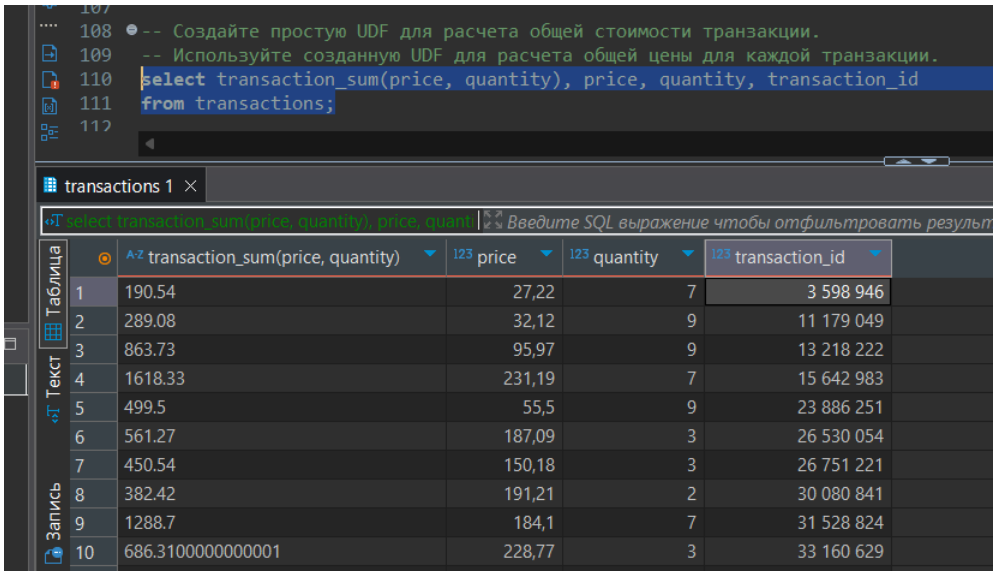
- Преобразуйте `transaction_id` в строку.

select toString(transaction_id), transaction_id
from transactions



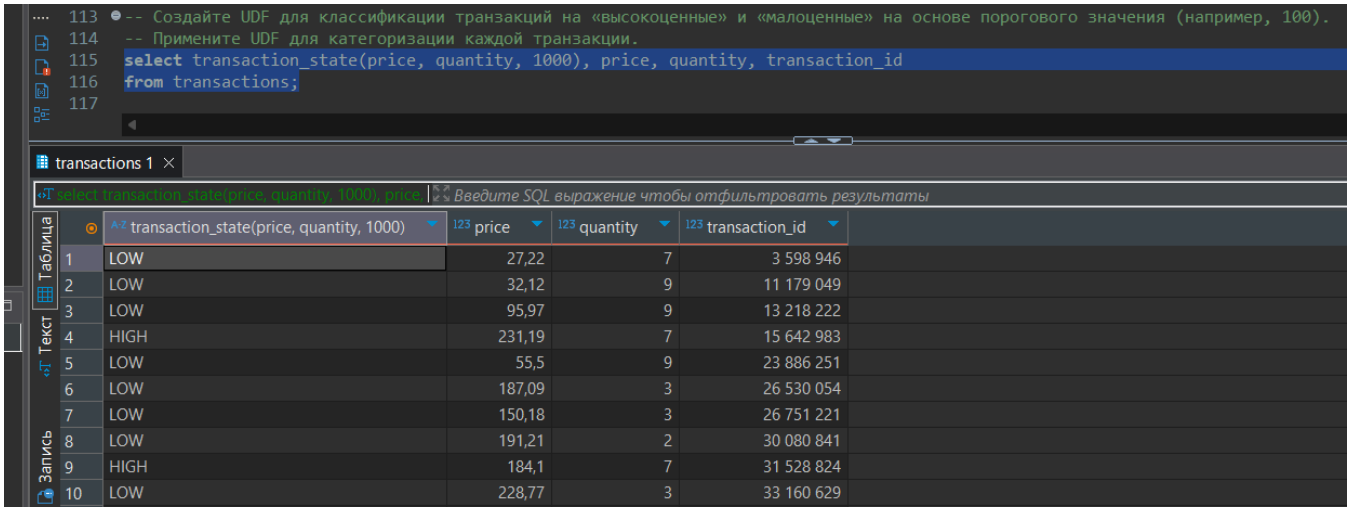

- Создайте простую UDF для расчета общей стоимости транзакции. Используйте созданную UDF для расчета общей цены для каждой транзакции.

select transaction_sum(price, quantity), price, quantity, transaction_id
from transactions



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

select transaction_state(price, quantity, 1000), price, quantity, transaction_id
from transactions;




**Создание UDF.**

Для использования своих функций нужно:

- добавить конфиг config.xml в папку /etc/clickhouse-server/config.d с содержимым:

```
<clickhouse>
        <user_defined_executable_functions_config>*.xml</user_defined_executable_functions_config>
</clickhouse>
```

- добавить xml с описанием функций в папку /etc/clickhouse-server :

- transaction_state.xml

```
<functions>
   <function>
      <type>executable</type>
      <name>transaction_state</name>
      <return_type>String</return_type>
      <argument>
         <type>Float64</type>
         <name>price</name>
      </argument>
      <argument>
         <type>UInt64</type>
         <name>quantity</name>
      </argument>
                  <argument>
         <type>UInt64</type>
         <name>limit</name>
      </argument>
      <format>TabSeparated</format>
      <command>transaction_state.py</command>
      <execute_direct>1</execute_direct>
   </function>
</functions>
```

- transaction_sum.xml

```
<functions>
   <function>
      <type>executable</type>
      <name>transaction_sum</name>
      <return_type>String</return_type>
      <argument>
         <type>Float64</type>
         <name>price</name>
      </argument>
      <argument>
         <type>UInt64</type>
         <name>quantity</name>
      </argument>
      <format>TabSeparated</format>
      <command>transaction_sum.py</command>
      <execute_direct>1</execute_direct>
   </function>
</functions>
```

- положить сами скрипты, написанные на python, в папку /var/lib/clickhouse/user_scripts

- transaction_state.py:

```
#!/usr/bin/python3
import sys

if __name__ == '__main__':
  for line in sys.stdin:
    arg1, arg2, arg3 = line.split('\t')

    result = '';
    if (float(arg1) * int(arg2)) > int(arg3):
      result = "HIGH"     # Высокоценная
    else:
      result = "LOW"      # Малоценная
    print(result)

    sys.stdout.flush()
```

- transaction_sum.py:

```
#!/usr/bin/python3
import sys

if __name__ == '__main__':
  for line in sys.stdin:
    arg1, arg2 = line.split('\t')

    result = float(arg1) * int(arg2)
    print(str(result))

    sys.stdout.flush()
```

Эти манипуляции происходят при сборке docker контейнера. Так же скриптам выставляются флаги для запуска:

```
chmod +x /var/lib/clickhouse/user_scripts/transaction_state.py
chmod +x /var/lib/clickhouse/user_scripts/transaction_sum.py
```

После запуска контейнера можно проверить, подтянулись ли новые функции. Нужно выполнить запрос: