**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"]

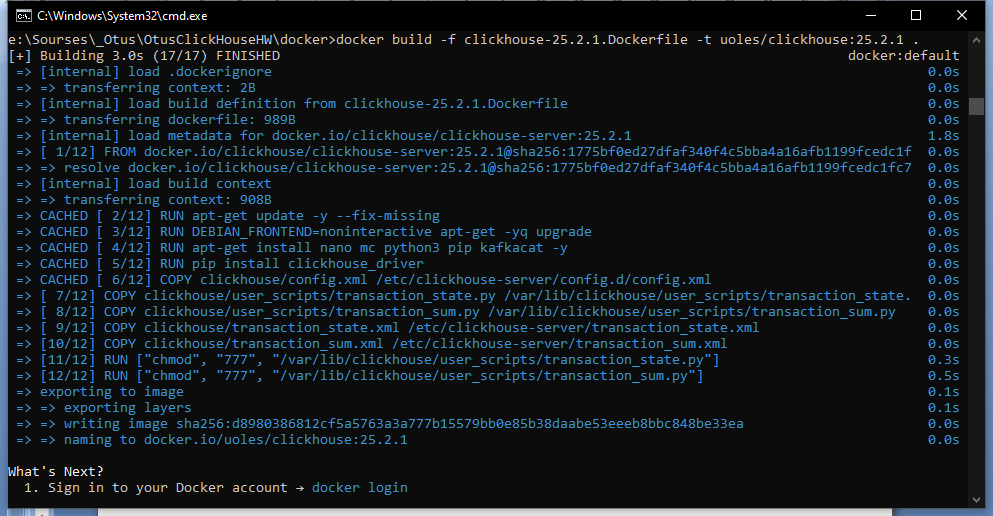
RUN ["chmod", "+x", "/var/lib/clickhouse/user\_scripts/transaction\_sum.py"]

EXPOSE 8123 9000

ENTRYPOINT ["/entrypoint.sh"]

Переходим в папку docker и cобираем 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 ^

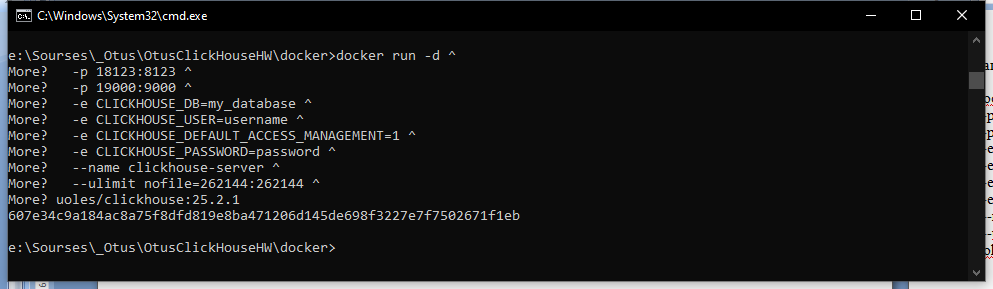
-e CLICKHOUSE\_DEFAULT\_ACCESS\_MANAGEMENT=1 ^

-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,

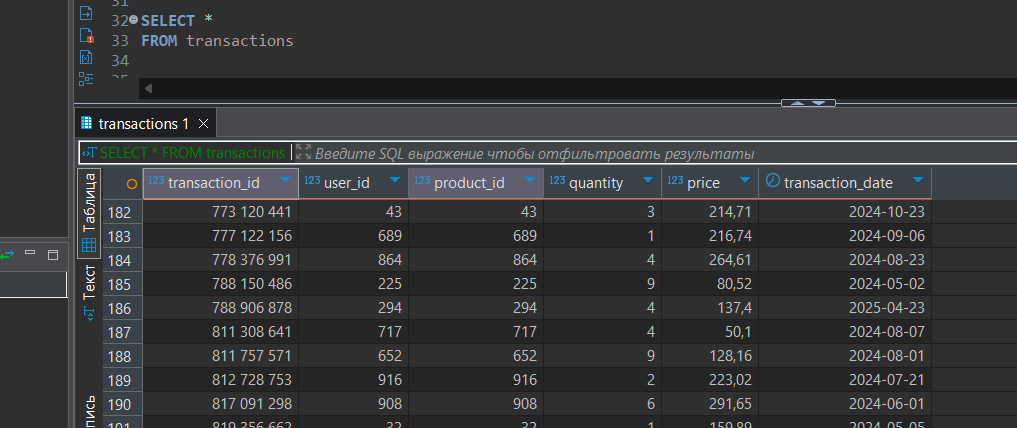
randUniform(1,10)::Int AS cnt,

round( randUniform(15.5, 299.99), 2 ) AS price,

now() - toIntervalSecond(rand() % (365 \* 24 \* 60 \* 60)) AS datetime

FROM system.numbers

LIMIT 1000;

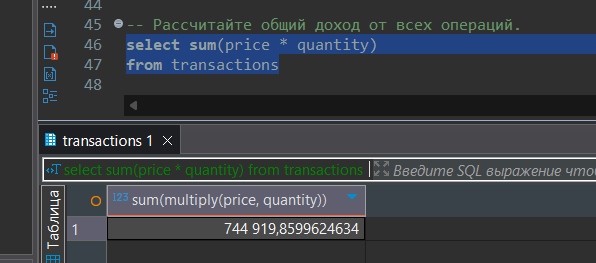


**Составляем запросы:**

* Рассчитайте общий доход от всех операций.

select sum(price \* quantity)

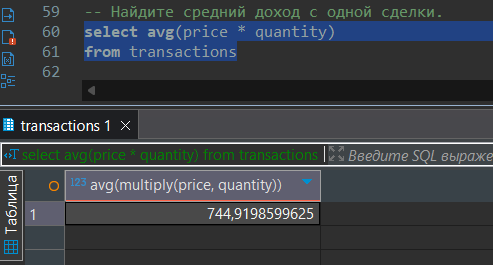
from transactions



* Найдите средний доход с одной сделки.

select avg(price \* quantity)

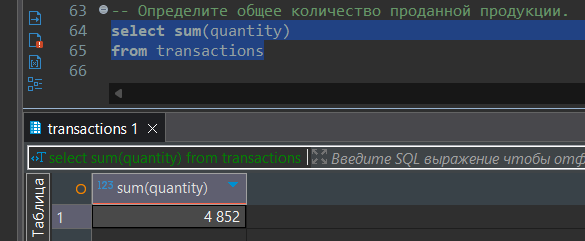
from transactions



* Определите общее количество проданной продукции.

select sum(quantity)

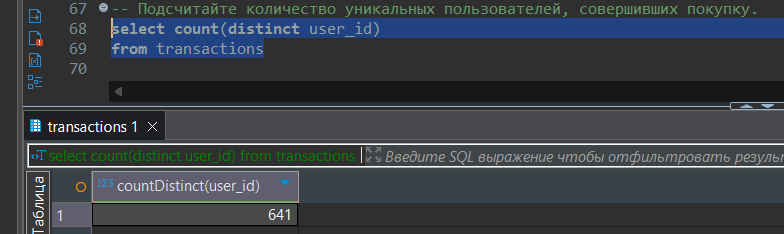
from transactions



* Подсчитайте количество уникальных пользователей, совершивших покупку.

select count(distinct user\_id)

from transactions

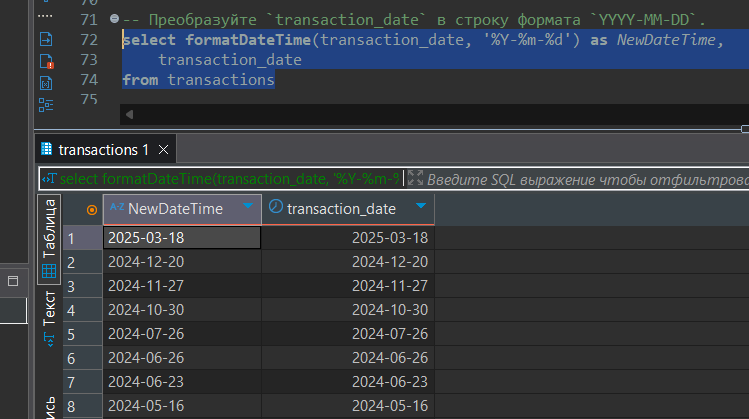


* Преобразуйте `transaction\_date` в строку формата `YYYY-MM-DD`.

select formatDateTime(transaction\_date, '%Y-%m-%d') as NewDateTime,

transaction\_date

from transactions



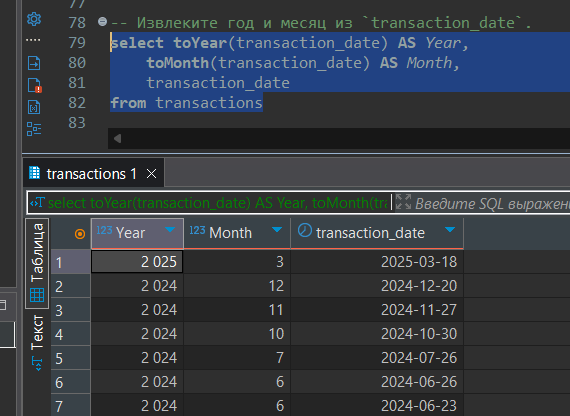
* Извлеките год и месяц из `transaction\_date`.

select toYear(transaction\_date) AS Year,

toMonth(transaction\_date) AS Month,

transaction\_date

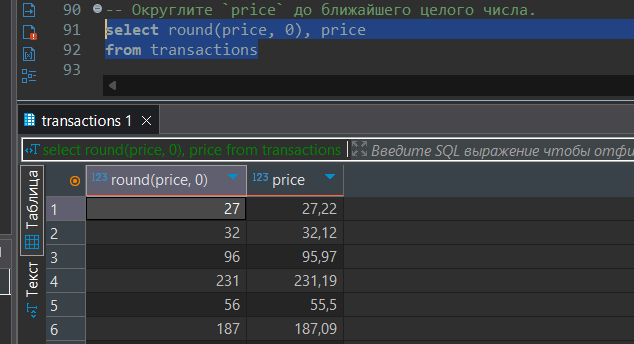
from transactions



* Округлите `price` до ближайшего целого числа.

select round(price, 0), price

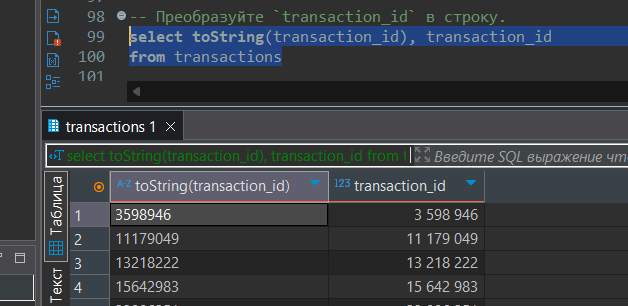
from transactions



* Преобразуйте `transaction\_id` в строку.

select toString(transaction\_id), transaction\_id

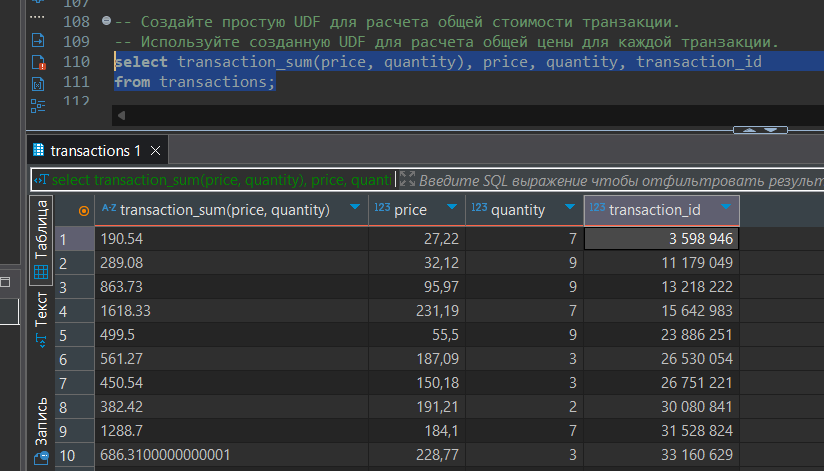
from transactions



* 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

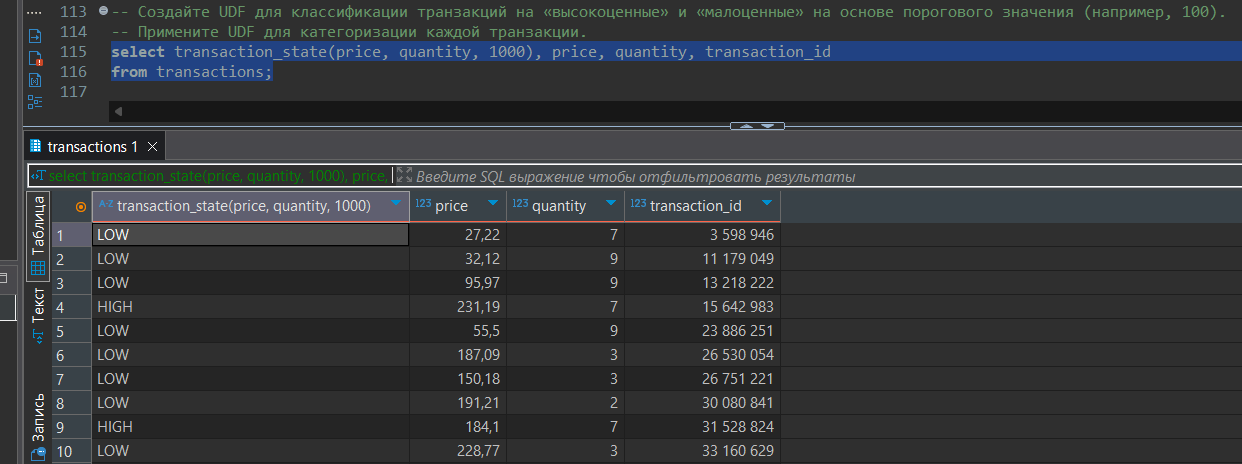
from transactions



* 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:

