Отчет к ДЗ по теме UDF, агрегатные функции и работа с типами данных

Создаем таблицу:

```
NB-R911LJCT.o3.ru :)  CREATE TABLE otus.transactio

CREATE TABLE otus.transactions
(
    `transaction_id` UInt32,
    `user_id` UInt32,
    `product_id` UInt32,
    `quantity` UInt8,
    `price` Float32,
    `transaction_date` Date
)
ENGINE = MergeTree
ORDER BY transaction_id

Query id: e9bce355-14bb-4e27-b09d-d2086c58b1cb

Ok.

0 rows in set. Elapsed: 0.018 sec.

NB-R911LJCT.o3.ru :)
```

Наполняем случайными данными

```
INSERT INTO otus.transactions FORMAT Values

Query id: d6b803c0-a103-45e4-909e-03475ff28de6

Ok.

3 rows in set. Elapsed: 0.009 sec.

NB-R911LJCT.o3.ru :)
```

```
SELECT *
FROM otus.transactions

Query id: e9d43507-229a-40c9-93d8-4bf2f2f13c52

    ┌─transaction_id─┬─user_id─┬─product_id─┬─quantity─┬─price─┬─transaction_date─┐
1.  │              1 │       1 │          1 │        3 │   100 │       2025-01-01 │
2.  │              2 │       1 │          2 │        1 │  1200 │       2025-01-02 │
3.  │              3 │     124 │          1 │        6 │   100 │       2025-01-03 │
    └────────────────┴─────────┴────────────┴──────────┴───────┴──────────────────┘

3 rows in set. Elapsed: 0.003 sec.

NB-R911LJCT.o3.ru :)
```

1. **Агрегатные функции**
   1.1. Рассчитайте общий доход от всех операций.

```
SELECT SUM(quantity * price) AS total_revenue
FROM otus.transactions

Query id: 0cae34b1-8c25-45fa-8cef-62cff3ed1dfb

    ┌─total_revenue─┐
1.  │          2100 │
    └───────────────┘

1 row in set. Elapsed: 0.045 sec.

NB-R911LJCT.o3.ru :)
```

   1.2. Найдите средний доход с одной сделки.

```
SELECT avg(quantity * price) AS average_revenue_per_transaction
FROM otus.transactions

Query id: 2be4c282-4707-48a0-be4f-a641e0c5b4e8

    ┌─average_revenue_per_transaction─┐
1.  │                             700 │
    └─────────────────────────────────┘

1 row in set. Elapsed: 0.043 sec.

NB-R911LJCT.o3.ru :)
```

   1.3. Определите общее количество проданной продукции.

```
NB-R911LJCT.o3.ru :) SELECT sum(quantity) AS total_quantity_sold FROM otus.transactions;

SELECT sum(quantity) AS total_quantity_sold
FROM otus.transactions

Query id: e0ab6317-fec3-4cfa-bfce-3d157e23d503

    ┌─total_quantity_sold─┐
1.  │                  10 │
    └─────────────────────┘

1 row in set. Elapsed: 0.005 sec.

NB-R911LJCT.o3.ru :)
```

1.4. Подсчитайте количество уникальных пользователей, совершивших покупку.

```
NB-R911LJCT.o3.ru :) SELECT uniq(user_id) AS unique_users FROM otus.transactions;

SELECT uniq(user_id) AS unique_users
FROM otus.transactions

Query id: 077ae3cb-1ec7-4b93-9af3-e20fad8cc6e3

   ┌─unique_users─┐
1. │            2 │
   └──────────────┘

1 row in set. Elapsed: 0.004 sec.

NB-R911LJCT.o3.ru :)
```

2. Функции для работы с типами данных

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

```
NB-R911LJCT.o3.ru :) SELECT formatDateTime(transaction_date, '%Y-%m-%d') AS t

SELECT formatDateTime(transaction_date, '%Y-%m-%d') AS transaction_date_str
FROM otus.transactions

Query id: f9daa69c-c87b-48ba-a872-594597e072ef

   ┌─transaction_date_str─┐
1. │ 2025-01-01           │
2. │ 2025-01-02           │
3. │ 2025-01-03           │
   └──────────────────────┘

3 rows in set. Elapsed: 0.003 sec.

NB-R911LJCT.o3.ru :)
```

2.2. Извлеките год и месяц из `transaction_date`.

```
SELECT
    toYear(transaction_date) AS transaction_year,
    toMonth(transaction_date) AS transaction_month
FROM otus.transactions

Query id: 06ebe6a4-855e-4f9b-8451-b61eb83f6de5

   ┌─transaction_year─┬─transaction_month─┐
1. │             2025 │                 1 │
2. │             2025 │                 1 │
3. │             2025 │                 1 │
   └──────────────────┴───────────────────┘

3 rows in set. Elapsed: 0.003 sec.

NB-R911LJCT.o3.ru :)
```

2.3. Округлите `price` до ближайшего целого числа.

```
SELECT round(price) AS rounded_price
FROM otus.transactions

Query id: dd69d745-ad74-4988-a167-46fe6ff3a40a

    ┌─rounded_price─┐
1.  │           100 │
2.  │          1200 │
3.  │           100 │
    └───────────────┘

3 rows in set. Elapsed: 0.002 sec.

NB-R911LJCT.o3.ru :)
```

2.4. Преобразуйте `transaction_id` в строку.

```
SELECT toString(transaction_id) AS transaction_id_str
FROM otus.transactions

Query id: 6531a589-171e-45cc-9218-a5fe763656bf

    ┌─transaction_id_str─┐
1.  │ 1                  │
2.  │ 2                  │
3.  │ 3                  │
    └────────────────────┘

3 rows in set. Elapsed: 0.002 sec.

NB-R911LJCT.o3.ru :)
```

3. User-Defined Functions (UDFs)

3.1. Создайте простую UDF для расчета общей стоимости транзакции.

```
NB-R911LJCT.o3.ru :) CREATE FUNCTION calculate_total_cost AS (x, y) -> x * y;

CREATE FUNCTION calculate_total_cost AS (x, y) -> (x * y)

Query id: b5b00e9c-2c4e-4624-9110-c2647768a6d7

Ok.

0 rows in set. Elapsed: 0.044 sec.

NB-R911LJCT.o3.ru :)
```

3.2. 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 otus.transactions

Query id: 3ea87a2e-e876-4d80-9039-0d9c7390b20a

    ┌─transaction_id─┬─user_id─┬─product_id─┬─quantity─┬─price─┬─total_cost─┐
1.  │              1 │       1 │          1 │        3 │   100 │        300 │
2.  │              2 │       1 │          2 │        1 │  1200 │       1200 │
3.  │              3 │     124 │          1 │        6 │   100 │        600 │
    └────────────────┴─────────┴────────────┴──────────┴───────┴────────────┘

3 rows in set. Elapsed: 0.002 sec.
```

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

```
CREATE FUNCTION classify_transaction AS x -> if(x > 100, 'high-value', 'low-value')

Query id: 7099857e-2d9e-4353-ada2-e79fa670a1b2

Ok.

0 rows in set. Elapsed: 0.018 sec.

NB-R911LJCT.o3.ru :)
```

3.4. 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(calculate_total_cost(quantity, price)) AS transaction_category
FROM otus.transactions

Query id: d6b61315-b90d-4bd0-9116-4200a4daabf1

    ┌─transaction_id─┬─user_id─┬─product_id─┬─quantity─┬─price─┬─total_cost─┬─transaction_category─┐
1.  │              1 │       1 │          1 │        3 │   100 │        300 │ high-value           │
2.  │              2 │       1 │          2 │        1 │  1200 │       1200 │ high-value           │
3.  │              3 │     124 │          1 │        6 │   100 │        600 │ high-value           │
    └────────────────┴─────────┴────────────┴──────────┴───────┴────────────┴──────────────────────┘

3 rows in set. Elapsed: 0.003 sec.

NB-R911LJCT.o3.ru :)
```