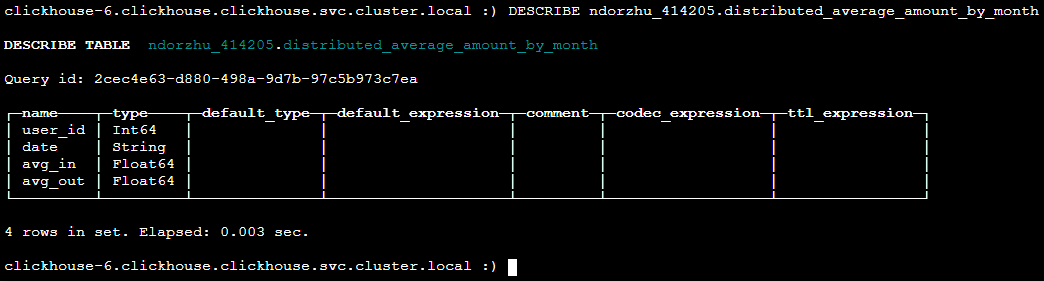
**Clickhouse lab**

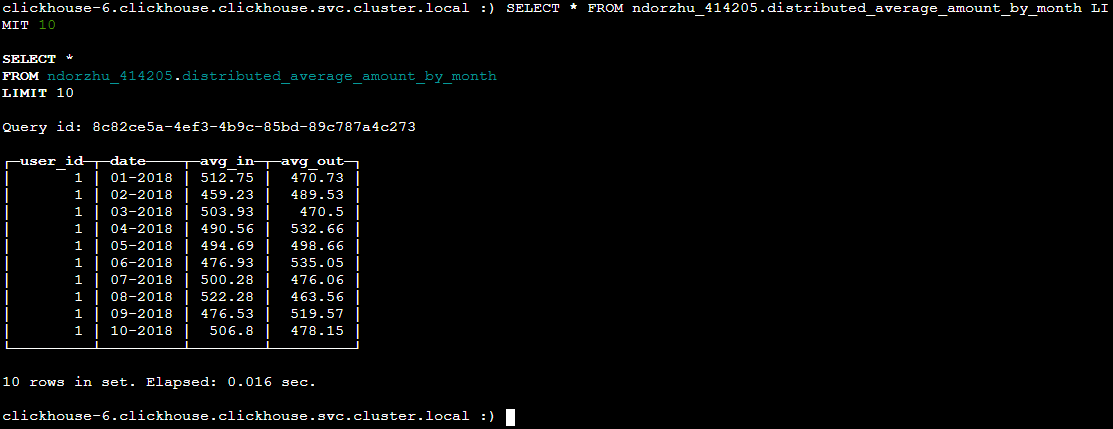
Выполнил: Доржу Начын Шолбанович, J4140, ndorzhu\_414205.

1. Средняя сумма входящих и исходящих транзакций по месяцам и дням для каждого пользователя.

**По месяцам**

1. Создал материализованное представление average\_amount\_by\_month и распределенную таблицу distributed\_average\_amount\_by\_month, которая ссылается на представление.

****

****

* user\_id - Идентификатор пользователя.
* Date - Месяц и год в формате MM-YYYY, к которым относятся средние значения транзакций.
* avg\_in - Средняя сумма входящих транзакций для данного пользователя в указанный месяц.
* avg\_out - Средняя сумма исходящих транзакций для данного пользователя в указанный месяц.

1. Обоснование выбора шардирующего выражения

Поле user\_id уникально для каждой записи и часто используется в запросах как основной ключ для агрегации данных. В случае, если система обслуживает миллионы пользователей, распределение данных по user\_id позволит равномерно распределять запросы. Например, данные будут параллельно обрабатываться на всех узлах кластера, что значительно ускорит выполнение запросов.

1. Перечислите выбранные MVS и укажите запросы на их создание.

CREATE MATERIALIZED VIEW ndorzhu\_414205.average\_amount\_by\_month

ON CLUSTER kube\_clickhouse\_cluster

ENGINE = AggregatingMergeTree

ORDER BY (user\_id, date) AS

WITH count\_in AS (

SELECT

user\_id\_in AS user\_id,

formatDateTime(datetime, '%m-%G') AS date,

ROUND(AVG(amount), 2) AS avg\_in

FROM ndorzhu\_414205.distributed\_users\_transactions

GROUP BY

user\_id,

date

),

count\_out AS (

SELECT

user\_id\_out AS user\_id,

formatDateTime(datetime, '%m-%G') AS date,

ROUND(AVG(amount), 2) AS avg\_out

FROM ndorzhu\_414205.distributed\_users\_transactions

GROUP BY

user\_id,

date

)

SELECT

tci.user\_id AS user\_id,

tci.date,

avg\_in,

avg\_out

FROM count\_in AS tci

INNER JOIN count\_out AS tco ON (tco.user\_id = tci.user\_id) AND (tco.date = tci.date)

ORDER BY

tci.user\_id,

date;

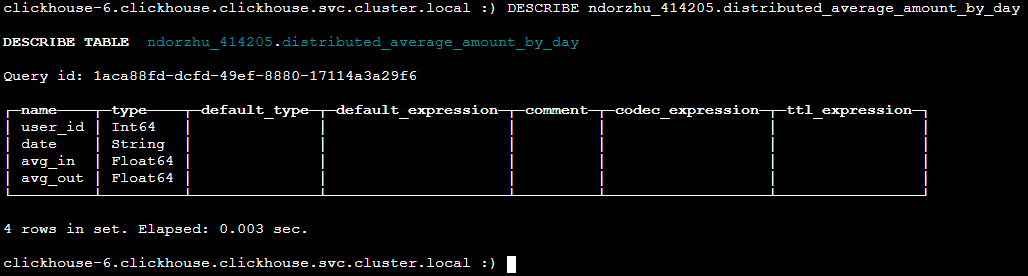
CREATE TABLE ndorzhu\_414205.distributed\_average\_amount\_by\_month

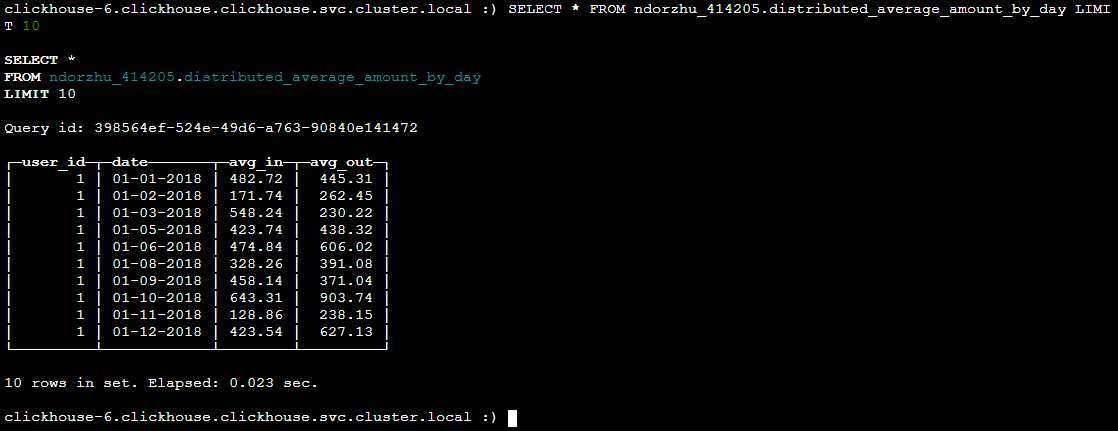
ON CLUSTER kube\_clickhouse\_cluster AS ndorzhu\_414205.average\_amount\_by\_month

ENGINE = Distributed(kube\_clickhouse\_cluster, ndorzhu\_414205, average\_amount\_by\_month);

**По дням**

1. Создал материализованное представление average\_amount\_by\_day и распределенную таблицу distributed\_average\_amount\_by\_day, которая ссылается на представление.





* user\_id - Идентификатор пользователя.
* Date - Дата, за которую рассчитывается средняя сумма транзакций.
* avg\_in - Средняя сумма входящих транзакций пользователя за указанный день.
* avg\_out - Средняя сумма исходящих транзакций пользователя за указанный день.

1. Обоснование выбора шардирующего выражения

Использование user\_id дает равномерное распределение нагрузки между узлами кластера, что помогает предотвратить перегрузку отдельных узлов. Такой подход облегчает масштабирование системы путем добавления новых узлов кластера без значительной переработки данных.

1. Перечислите выбранные MVS и укажите запросы на их создание.

CREATE MATERIALIZED VIEW ndorzhu\_414205.average\_amount\_by\_day

ON CLUSTER kube\_clickhouse\_cluster

ENGINE = AggregatingMergeTree

ORDER BY (user\_id, date) AS

WITH count\_in AS (

SELECT

user\_id\_in AS user\_id,

formatDateTime(datetime, '%d-%m-%G') AS date,

ROUND(AVG(amount), 2) AS avg\_in

FROM ndorzhu\_414205.distributed\_users\_transactions

GROUP BY

user\_id,

date

),

count\_out AS (

SELECT

user\_id\_out AS user\_id,

formatDateTime(datetime, '%d-%m-%G') AS date,

ROUND(AVG(amount), 2) AS avg\_out

FROM ndorzhu\_414205.distributed\_users\_transactions

GROUP BY

user\_id,

date

)

SELECT

tci.user\_id AS user\_id,

tci.date,

avg\_in,

avg\_out

FROM count\_in AS tci

INNER JOIN count\_out AS tco ON (tco.user\_id = tci.user\_id) AND (tco.date = tci.date)

ORDER BY

tci.user\_id,

date;

CREATE TABLE ndorzhu\_414205.distributed\_average\_amount\_by\_day

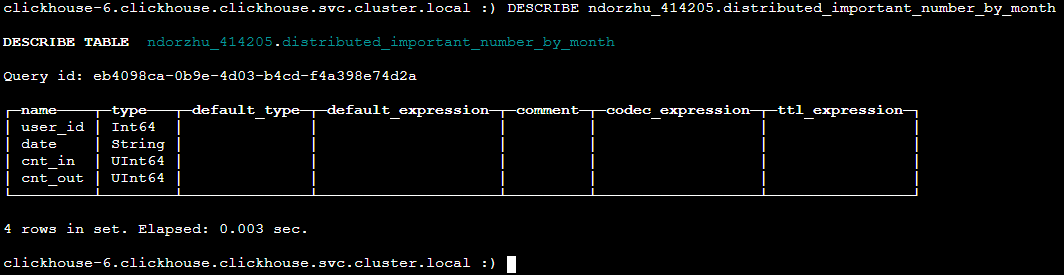
ON CLUSTER kube\_clickhouse\_cluster AS ndorzhu\_414205.average\_amount\_by\_day

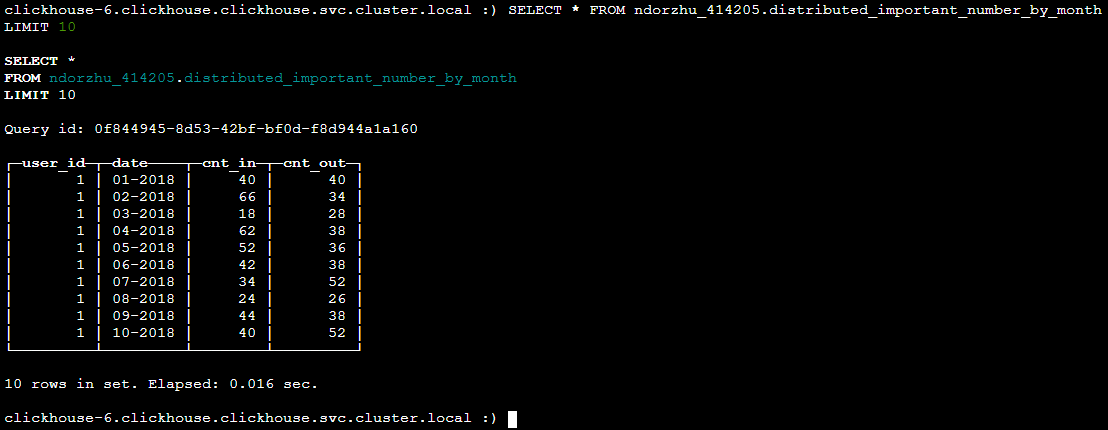
ENGINE = Distributed(kube\_clickhouse\_cluster, ndorzhu\_414205, average\_amount\_by\_day);

1. Количество важных транзакций для входящих и исходящих транзакций по месяцам и дням для каждого пользователя.

**По месяцам**

1. Создал материализованное представление important\_number\_by\_month и распределенную таблицу distributed\_important\_number\_by\_month, которая ссылается на представление.





* user\_id - Идентификатор пользователя.
* date - Месяц и год, за который рассчитывается количество транзакций важного типа.
* cnt\_in - Количество важных входящих транзакций для пользователя за указанный месяц.
* cnt\_out - Количество важных исходящих транзакций для пользователя за указанный месяц.

1. Обоснование выбора шардирующего выражения

Поле user\_id идентифицирует каждого пользователя, а также часто используется в запросах как основной ключ для агрегации данных. Использование user\_id обеспечивает равномерное распределение данных по кластерам. В случае обработки данных о транзакциях с учетом их важности, распределение данных по user\_id обеспечит эффективное выполнение запросов и анализа важных транзакций пользователей в различные периоды времени.

1. Перечислите выбранные MVS и укажите запросы на их создание.

CREATE MATERIALIZED VIEW ndorzhu\_414205.important\_number\_by\_month

ON CLUSTER kube\_clickhouse\_cluster

ENGINE = AggregatingMergeTree

ORDER BY (user\_id, date) AS

WITH count\_in AS (

SELECT

user\_id\_in AS user\_id,

formatDateTime(datetime, '%m-%G') AS date,

COUNT(amount) AS cnt\_in

FROM ndorzhu\_414205.distributed\_users\_transactions

WHERE important = 1

GROUP BY

user\_id,

date

),

count\_out AS (

SELECT

user\_id\_out AS user\_id,

formatDateTime(datetime, '%m-%G') AS date,

COUNT(amount) AS cnt\_out

FROM ndorzhu\_414205.distributed\_users\_transactions

WHERE important = 1

GROUP BY

user\_id,

date

)

SELECT

tci.user\_id AS user\_id,

tci.date AS date,

cnt\_in,

cnt\_out

FROM count\_in AS tci

INNER JOIN count\_out AS tco ON (tco.user\_id = tci.user\_id) AND (tco.date = tci.date)

ORDER BY

date;

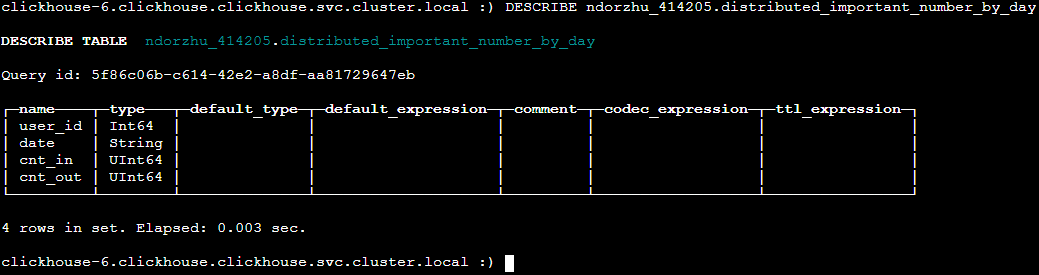
CREATE TABLE ndorzhu\_414205.distributed\_important\_number\_by\_month

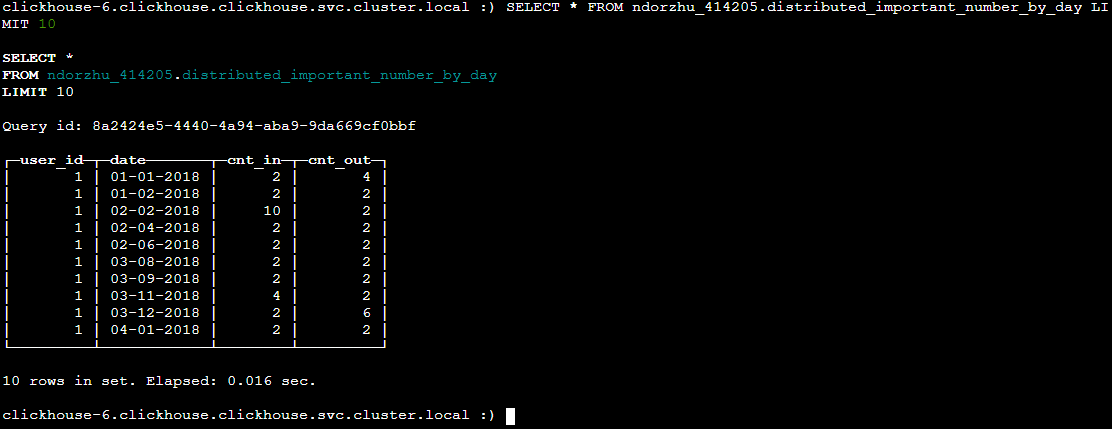
ON CLUSTER kube\_clickhouse\_cluster AS ndorzhu\_414205.important\_number\_by\_month

ENGINE = Distributed(kube\_clickhouse\_cluster, ndorzhu\_414205, important\_number\_by\_month);

**По дням**

1. Создал материализованное представление important\_number\_by\_day и распределенную таблицу distributed\_important\_number\_by\_day, которая ссылается на представление.





* user\_id - Идентификатор пользователя.
* date - Дата, за которую рассчитывается количество важных транзакций.
* cnt\_in - Количество важных входящих транзакций для пользователя за указанный день.
* cnt\_out - Количество важных исходящих транзакций для пользователя за указанный день.

1. Обоснование выбора шардирующего выражения

Выбор user\_id в качестве шардирующего выражения для создания материализованного представления и распределенной таблицы был логичным, с учетом производительности, масштабируемости и балансировки нагрузки. Этот подход позволяет эффективно обрабатывать данные и выполнять запросы в системах с высокой производительностью и масштабируемостью.

3) Перечислите выбранные MVS и укажите запросы на их создание.

CREATE MATERIALIZED VIEW ndorzhu\_414205.important\_number\_by\_day

ON CLUSTER kube\_clickhouse\_cluster

ENGINE = AggregatingMergeTree

ORDER BY (user\_id, date) AS

WITH count\_in AS (

SELECT

user\_id\_in AS user\_id,

formatDateTime(datetime, '%d-%m-%G') AS date,

COUNT(amount) AS cnt\_in

FROM ndorzhu\_414205.distributed\_users\_transactions

WHERE important = 1

GROUP BY

user\_id,

date

),

count\_out AS (

SELECT

user\_id\_out AS user\_id,

formatDateTime(datetime, '%d-%m-%G') AS date,

COUNT(amount) AS cnt\_out

FROM ndorzhu\_414205.distributed\_users\_transactions

WHERE important = 1

GROUP BY

user\_id,

date

)

SELECT

tci.user\_id AS user\_id,

tci.date AS date,

cnt\_in,

cnt\_out

FROM count\_in AS tci

INNER JOIN count\_out AS tco ON (tco.user\_id = tci.user\_id) AND (tco.date = tci.date)

ORDER BY

date;

CREATE TABLE ndorzhu\_414205.distributed\_important\_number\_by\_day

ON CLUSTER kube\_clickhouse\_cluster AS ndorzhu\_414205.important\_number\_by\_day

ENGINE = Distributed(kube\_clickhouse\_cluster, ndorzhu\_414205, important\_number\_by\_day);