[redacted]

1. Создадим таблицу для данных из задания

UInt32 подходит для хранения id пользователей, т.к. они являются целыми положительными числами, а UInt8 - для хранения бинарных значений из колонки important. Колонка amount содержит дробные числа, поэтому используем Float32.

```
CREATE TABLE [redacted].user_transactions ON CLUSTER [redacted]
(
    user_id_out UInt32,
    user_id_in UInt32,
    important UInt8,
    amount Float32,
    datetime DateTime
)
ENGINE = MergeTree()
PARTITION BY toYYYYMM(datetime)
ORDER BY (user_id_out, datetime, important);
```

2. Создадим распределенную таблицу

Во всех распределенных таблицах используется функция xxHash64 с ключем user_id или user_id_out для оптимизации работы запросов с JOIN и равномерного распределения данных по узлам. Также, функция xxHash64 является быстрой не-криптографической хеш-функцией и обеспечивает хорошее распределение данных в кластере.

```
CREATE TABLE [redacted].user_transactions_distributed ON CLUSTER [redacted] AS
[redacted].user_transactions
ENGINE = Distributed([redacted], [redacted], user_transactions,
xxHash64(user_id_out));
```

3. Создадим Materialized View

3.1. MV для подсчета среднего значения транзакций

Подсчет среднего значения исходящих транзакций

Создание Materialized View:

```
CREATE MATERIALIZED VIEW [redacted].avg_amount_out_mv ON CLUSTER [redacted]
ENGINE = AggregatingMergeTree()
ORDER BY (user_id, date)
AS SELECT
    user_id_out as user_id,
    formatDateTime(datetime, '%Y-%m') as date,
    avgState(amount) as avg_amount_out
FROM [redacted].user_transactions
GROUP BY user_id, date;
```

Создание распределенной таблицы:

```
CREATE TABLE [redacted].avg_amount_out_mv_distr ON CLUSTER [redacted] AS
[redacted].avg_amount_out_mv
ENGINE = Distributed([redacted], [redacted], avg_amount_out_mv,
xxHash64(user_id));
```

Подсчет среднего значения входящих транзакций

Создание Materialized View:

```
CREATE MATERIALIZED VIEW [redacted].avg_amount_in_mv ON CLUSTER [redacted]
ENGINE = AggregatingMergeTree()
ORDER BY (user_id, date)
AS SELECT
    user_id_in as user_id,
    formatDateTime(datetime, '%Y-%m') as date,
    avgState(amount) as avg_amount_in
FROM [redacted].user_transactions
GROUP BY user_id, date;
```

Создание распределенной таблицы:

```
CREATE TABLE [redacted].avg_amount_in_mv_distr ON CLUSTER [redacted] AS
```

```
[redacted].avg_amount_in_mv
ENGINE = Distributed([redacted], [redacted], avg_amount_in_mv, xxHash64(user_id));
```

VIEW для вывода результатов

Подсчет среднего значения исходящих транзакций:

```
CREATE VIEW [redacted].avg_amount_out_view ON CLUSTER [redacted]
AS SELECT
    user_id,
    avgMerge(avg_amount_out) as avg_amount_out,
    date
FROM [redacted].avg_amount_out_mv_distr
GROUP BY
    user_id,
    date
ORDER BY
    user_id,
    date
```

Подсчет среднего значения входящих транзакций:

```
CREATE VIEW [redacted].avg_amount_in_view ON CLUSTER [redacted]
AS SELECT
    user_id,
    avgMerge(avg_amount_in) as avg_amount_in,
    date
FROM [redacted].avg_amount_in_mv_distr
GROUP BY
    user_id,
    date
ORDER BY
    user_id,
    date
```

Совместное отображение:

```
CREATE VIEW [redacted].avg_amount_view ON CLUSTER [redacted]
AS SELECT
    user_id,
    avg_amount_out,
    avg_amount_in,
    date
FROM [redacted].avg_amount_out_view as l
LEFT JOIN [redacted].avg_amount_in_view as r on l.user_id=r.user_id and l.date=r.date
ORDER BY
    user_id,
    date
```

3.2. MV для подсчета количества важных транзакций

Подсчет количества важных исходящих транзакций

Создание Materialized View:

```
CREATE MATERIALIZED VIEW [redacted].important_count_out_mv ON CLUSTER [redacted]
ENGINE = AggregatingMergeTree()
ORDER BY (user_id, date)
AS SELECT
    user_id_out as user_id,
    countState(user_id) as important_count_out,
    formatDateTime(datetime, '%Y-%m') as date
FROM [redacted].user_transactions
GROUP BY user_id, date
HAVING important = 1;
```

Создание распределенной таблицы:

```
CREATE TABLE [redacted].important_count_out_mv_distr ON CLUSTER [redacted] AS
[redacted].important_count_out_mv
ENGINE = Distributed([redacted], [redacted], important_count_out_mv,
xxHash64(user_id));
```

Подсчет количества важных входящих транзакций

Создание Materialized View:

```
CREATE MATERIALIZED VIEW [redacted].important_count_in_mv ON CLUSTER [redacted]
ENGINE = AggregatingMergeTree()
```

```
ORDER BY (user_id, date)
AS SELECT
    user_id_in as user_id,
    countState(user_id) as important_count_in,
    formatDateTime(datetime, '%Y-%m') as date
FROM [redacted].user_transactions
GROUP BY user_id, date
HAVING important = 1;
```

Создание распределенной таблицы:

```
CREATE TABLE [redacted].important_count_in_mv_distr ON CLUSTER [redacted] AS
[redacted].important_count_in_mv
ENGINE = Distributed([redacted], [redacted], important_count_in_mv,
xxHash64(user_id));
```

VIEW для вывода результатов

Подсчет количества важных исходящих транзакций:

```
CREATE VIEW [redacted].important_count_out_view ON CLUSTER [redacted]
AS SELECT
    user_id,
    countMerge(important_count_out) as important_count_out,
    date
FROM [redacted].important_count_out_mv_distr
GROUP BY
    user_id,
    date
ORDER BY
    user_id,
    date
```

Подсчет количества важных входящих транзакций:

```
CREATE VIEW [redacted].important_count_in_view ON CLUSTER [redacted]
AS SELECT
    user_id,
    countMerge(important_count_in) as important_count_in,
    date
FROM [redacted].important_count_in_mv_distr
GROUP BY
    user_id,
    date
ORDER BY
    user_id,
    date
```

Совместное отображение:

```
CREATE VIEW [redacted].important_count_view ON CLUSTER [redacted]
AS SELECT
    user_id,
    important_count_out,
    important_count_in,
    date
FROM [redacted].important_count_out_view as l
LEFT JOIN [redacted].important_count_in_view as r on l.user_id=r.user_id and l.date=r.date
ORDER BY
    user_id,
    date
```

4. Скриншот с созданными таблицами

```
name
 .inner_id.4c979568-3afd-4390-8c97-95683afd8390
 .inner_id.a079754d-b2dd-4418-a079-754db2dd7418
 .inner_id.d4f7c202-8d1c-44ad-94f7-c2028d1ce4ad
 .inner_id.fab9d72d-cbff-4feb-bab9-d72dcbffdfeb
avg_amount_in_mv
avg_amount_in_mv_distr
avg_amount_in_view
avg_amount_out_mv
avg_amount_out_mv_distr
avg_amount_out_view
avg_amount_view
important_count_in_mv
important_count_in_mv_distr
important_count_in_view
important_count_out_mv
important_count_out_mv_distr
important_count_out_view
important_count_view
user_transactions
user_transactions_distributed
```

Таблицы .inner_id. появляются, т.к. наши Materialized Views не направляют обработанные данные в другую таблицу с помощью TO.

5. Отправим данные в ClickHouse

Отправка данных происходит в конце работы, чтобы убедиться в правильности работы таблиц.

```
cat transactions_12M.parquet | clickhouse-client --user=[redacted] --host=clickhouse-
4.clickhouse.clickhouse --password= --query="INSERT INTO
[redacted].user_transactions_distributed FORMAT Parquet"
```

6. Вывод результатов

```
SELECT *
FROM avg_amount_out_view
LIMIT 12

Query id: 88ee4258-9c6a-4fdc-bb16-a5dc748f6527

┌─user_id─┬──────avg_amount_out─┬─date───┐
│       1 │ 470.7286178861789   │ 2018-01 │
│       1 │ 489.5341666666667   │ 2018-02 │
│       1 │ 470.4968292682927   │ 2018-03 │
│       1 │ 532.6629292929292   │ 2018-04 │
│       1 │             498.6649 │ 2018-05 │
│       1 │  535.052982905983   │ 2018-06 │
│       1 │ 476.05547826086956  │ 2018-07 │
│       1 │ 463.55867058823530  │ 2018-08 │
│       1 │ 519.5729702970298   │ 2018-09 │
│       1 │ 478.14594949494950  │ 2018-10 │
│       1 │             480.9221 │ 2018-11 │
│       1 │ 492.6372526315789   │ 2018-12 │
└─────────┴─────────────────────┴────────┘
```

Подсчет среднего значения исходящих транзакций.

```
SELECT *
FROM avg_amount_in_view
LIMIT 12

Query id: d1c860d8-4619-4147-8aba-767996e5ed94
```

user_id	avg_amount_in	date
1	512.7485106382978	2018-01
1	459.2311779661017	2018-02
1	503.93045454545	2018-03
1	490.5564705882353	2018-04
1	494.69410784313726	2018-05
1	476.9311881188119	2018-06
1	500.28178723404255	2018-07
1	522.2833333333333	2018-08
1	476.5315	2018-09
1	506.80349	2018-10
1	531.8787610619469	2018-11
1	537.7928315789474	2018-12

Подсчет среднего значения входящих транзакций.

```
SELECT *
FROM avg_amount_view
LIMIT 12

Query id: 3a24b683-fedc-430e-9a95-a4286a56184f
```

user_id	avg_amount_out	avg_amount_in	date
1	470.7286178861789	512.7485106382978	2018-01
1	489.5341666666667	459.2311779661017	2018-02
1	470.4968292682927	503.93045454545	2018-03
1	532.6629292929292	490.5564705882353	2018-04
1	498.6649	494.69410784313726	2018-05
1	535.052982905983	476.9311881188119	2018-06
1	476.05547826086956	500.28178723404255	2018-07
1	463.5586705882353	522.2833333333333	2018-08
1	519.5729702970298	476.5315	2018-09
1	478.14594949495	506.80349	2018-10
1	480.9221	531.8787610619469	2018-11
1	492.6372526315789	537.7928315789474	2018-12

Совместное отображение для подсчета среднего значения транзакций.

```
SELECT *
FROM important_count_out_view
LIMIT 12

Query id: 60dc11b0-001e-414c-a091-74cdf772bc5e

┌─user_id─┬─important_count_out─┬─date───┐
│       1 │                  20 │ 2018-01 │
│       1 │                  17 │ 2018-02 │
│       1 │                  14 │ 2018-03 │
│       1 │                  19 │ 2018-04 │
│       1 │                  18 │ 2018-05 │
│       1 │                  19 │ 2018-06 │
│       1 │                  26 │ 2018-07 │
│       1 │                  13 │ 2018-08 │
│       1 │                  19 │ 2018-09 │
│       1 │                  26 │ 2018-10 │
│       1 │                  14 │ 2018-11 │
│       1 │                  24 │ 2018-12 │
└─────────┴─────────────────────┴─────────┘
```

Подсчет количества важных исходящих транзакций.

```
SELECT *
FROM important_count_in_view
LIMIT 12

Query id: b482f244-9c33-4e96-89d9-21f5990236f2

┌─user_id─┬─important_count_in─┬─date───┐
│       1 │                 20 │ 2018-01 │
│       1 │                 33 │ 2018-02 │
│       1 │                  9 │ 2018-03 │
│       1 │                 31 │ 2018-04 │
│       1 │                 26 │ 2018-05 │
│       1 │                 21 │ 2018-06 │
│       1 │                 17 │ 2018-07 │
│       1 │                 12 │ 2018-08 │
│       1 │                 22 │ 2018-09 │
│       1 │                 20 │ 2018-10 │
│       1 │                 23 │ 2018-11 │
│       1 │                 19 │ 2018-12 │
└─────────┴────────────────────┴─────────┘
```

Подсчет количества важных входящих транзакций.

```
SELECT *
FROM important_count_view
LIMIT 12

Query id: bf1e9e14-d917-4b66-82c4-7d5a98f87395

┌─user_id─┬─important_count_out─┬─important_count_in─┬─date────┐
│       1 │                  20 │                 20 │ 2018-01 │
│       1 │                  17 │                 33 │ 2018-02 │
│       1 │                  14 │                  9 │ 2018-03 │
│       1 │                  19 │                 31 │ 2018-04 │
│       1 │                  18 │                 26 │ 2018-05 │
│       1 │                  19 │                 21 │ 2018-06 │
│       1 │                  26 │                 17 │ 2018-07 │
│       1 │                  13 │                 12 │ 2018-08 │
│       1 │                  19 │                 22 │ 2018-09 │
│       1 │                  26 │                 20 │ 2018-10 │
│       1 │                  14 │                 23 │ 2018-11 │
│       1 │                  24 │                 19 │ 2018-12 │
└─────────┴─────────────────────┴────────────────────┴─────────┘
```

Совместное отображение для подсчета количества важных транзакций.

7. Список команд для вывода результатов

Команда для вывода среднего значения транзакций каждого пользователя:

```
select * from [redacted].avg_amount_view
```

Команда для вывода количества важных транзакций каждого пользователя:

```
select * from [redacted].important_count_view
```

8. Проверка результатов

Проверка результатов проводилась сравнением результатов из View с запросами из изначальной таблицы. Все результаты сошлись. Например, для подсчета важных транзакций:

Проверка подсчета входящих важных транзакций:

```
SELECT
    user_id_in,
    count(user_id_in) AS count,
    formatDateTime(datetime, '%Y-%m') AS date
FROM user_transactions_distributed
GROUP BY
    user_id_in,
    date
HAVING important = 1
ORDER BY
    user_id_in ASC,
    date ASC
LIMIT 12
```

Проверка подсчета исходящих важных транзакций:

```
SELECT
    user_id_out,
    count(user_id_out) AS count,
    formatDateTime(datetime, '%Y-%m') AS date
FROM user_transactions_distributed
GROUP BY
    user_id_out,
    date
HAVING important = 1
ORDER BY
    user_id_out ASC,
    date ASC
LIMIT 12
```

```
SELECT
    user_id_out,
    count(user_id_out) AS count,
    formatDateTime(datetime, '%Y-%m') AS date
FROM user_transactions_distributed
GROUP BY
    user_id_out,
    date
HAVING important = 1
ORDER BY
    user_id_out ASC,
    date ASC
LIMIT 12

Query id: 312e32d9-c125-4ecf-a82f-741ede75739b

┌─user_id_out─┬─count─┬─date────┐
│           1 │    20 │ 2018-01 │
│           1 │    17 │ 2018-02 │
│           1 │    14 │ 2018-03 │
│           1 │    19 │ 2018-04 │
│           1 │    18 │ 2018-05 │
│           1 │    19 │ 2018-06 │
│           1 │    26 │ 2018-07 │
│           1 │    13 │ 2018-08 │
│           1 │    19 │ 2018-09 │
│           1 │    26 │ 2018-10 │
│           1 │    14 │ 2018-11 │
│           1 │    24 │ 2018-12 │
└─────────────┴───────┴─────────┘
```

Проверка подсчета исходящих важных транзакций.

```
SELECT
    user_id_in,
    count(user_id_in) AS count,
    formatDateTime(datetime, '%Y-%m') AS date
FROM user_transactions_distributed
GROUP BY
    user_id_in,
    date
HAVING important = 1
ORDER BY
    user_id_in ASC,
    date ASC
LIMIT 12

Query id: 5328efd5-ee36-4181-b209-c43e8d3b73fb

┌─user_id_in─┬─count─┬─date────┐
│          1 │    20 │ 2018-01 │
│          1 │    33 │ 2018-02 │
│          1 │     9 │ 2018-03 │
│          1 │    31 │ 2018-04 │
│          1 │    26 │ 2018-05 │
│          1 │    21 │ 2018-06 │
│          1 │    17 │ 2018-07 │
│          1 │    12 │ 2018-08 │
│          1 │    22 │ 2018-09 │
│          1 │    20 │ 2018-10 │
│          1 │    23 │ 2018-11 │
│          1 │    19 │ 2018-12 │
└────────────┴───────┴─────────┘
```

Проверка подсчета входящих важных транзакций.