ClickHouse Lab

During this lab you have to implement Data Warehouse (DWH) using ClickHouse (CH) and its techniques such as Materialized View (MV) and Distributed tables.

Dataset

Dataset is presented by a parquet file with users' transactions. Path to this file:
`/nfs/shared/clickhouse_data/transactions_12M.parquet`

Dataset sample:

user_id_out	user_id_in	important	amount	datetime
2781	3343	0	199.2	2018-09-02 17:25:12
2789	3343	0	566.33	2018-11-26 11:29:26
2838	3343	0	85.42	2018-09-05 19:59:22
2850	3343	0	850.74	2018-02-19 14:47:41
2860	3343	0	238.35	2018-10-16 00:58:21
2872	3343	0	940.16	2018-09-17 08:24:18
2874	3343	0	308	2018-12-13 11:59:50
2878	3343	0	709.32	2018-11-20 10:35:57
2891	3343	0	121.71	2018-11-27 03:59:52
2939	3343	0	240.06	2018-08-27 20:03:52

Dataset properties:

- ~20% transactions are important (important == 1)
- Total records amount - 12 millions

Task

1. You have to choose **2 or more** MVs. The MVs list is located below.

    a. **Users' saldo for the current moment.**
    b. **The sums for incoming and outcoming transactions by months for each user.**

2. Upload the data into the CH cluster. Table for uploaded data has to be the MergeTree family. To distribute data over the cluster you have to use the Distributed engine and sharding expression.

```
CREATE TABLE prodriges_374222.transacciones_bueno ON CLUSTER
kube_clickhouse_cluster (
    `user_id_out` Int64,
    `user_id_in` Int64,
    `important` Int8,
    `amount` Float64,
    `datetime` DateTime
) ENGINE = MergeTree
PARTITION BY toYYYYMM(datetime)
ORDER BY user_id_out;
```

Justification:

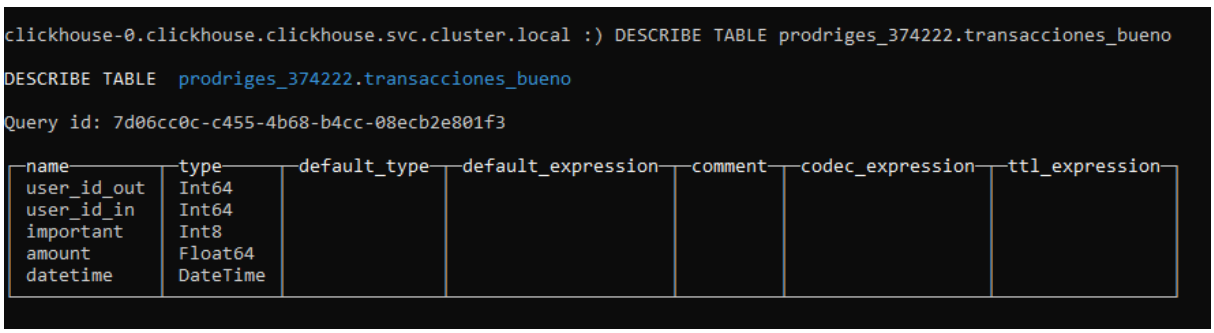
- **Partitioned by toYYYYMM(datetime):**
  - **Reason:** This divides the data into monthly partitions, optimizing time-based queries and making it easier to manage large datasets.
- **Ordered by (user_id_out):**
  - **Reason:** Ordering the data by user_id_out optimizes queries that filter or aggregate data based on user IDs. This ordering enhances the performance of queries that focus on specific users, such as retrieving all transactions for a particular user or calculating aggregate statistics like total transaction amounts per user. This order also improves data compression, further enhancing query efficiency and storage utilization.

```
CREATE TABLE prodriges_374222.distribuido_transacciones_bueno ON CLUSTER
kube_clickhouse_cluster AS prodriges_374222.transacciones_bueno
ENGINE = Distributed(kube_clickhouse_cluster, prodriges_374222,
transacciones_bueno, xxHash64(user_id_out));
```

Justification:

- **Sharding by xxHash64(datetime):**
  - **Explanation:** Sharding by `xxHash64(user_id_out)` distributes the data evenly across the nodes in the cluster based on the hash of the `user_id_out` value. This ensures that the data is distributed in a balanced manner, preventing any single node from becoming a bottleneck (hot spot). By evenly distributing the data, this approach improves the overall query performance and resource utilization of the cluster.

```
DESCRIBE TABLE prodriges_374222.transacciones_bueno
```

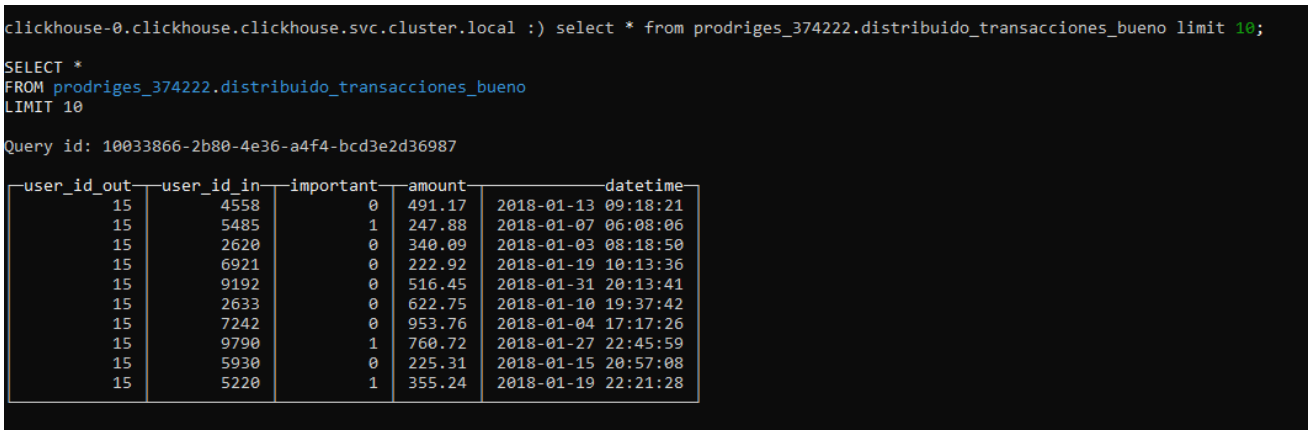



```
cat shared-data/clickhouse_data/transactions_12M.parquet | \
clickhouse-client --host=clickhouse-0.clickhouse.clickhouse \
--user=prodriges_374222 \
--password=479Ak98oRi \
--query="INSERT INTO prodriges_374222. distribuido_transacciones_bueno
FORMAT Parquet"
```

Justification:

- • **cat shared-data/clickhouse_data/transactions_12M.parquet:**
  - **Reason:** This command reads the file transactions_12M.parquet which contains the data to be inserted into the distributed_user_transac table.
- **clickhouse-client --host=clickhouse-0.clickhouse.clickhouse \--user=prodriges_374222 \--password=479Ak98oRi \--query="INSERT INTO prodriges_374222. distribuido_transacciones_bueno FORMAT Parquet"**
  - **Reason:** Utilizes the ClickHouse client to insert data into the distribuido_transacciones_bueno table in the specified cluster.

```
SELECT * FROM prodriges_374222. distribuido_transacciones_bueno limit 10
```



```
clickhouse-0.clickhouse.clickhouse.svc.cluster.local :) select * from prodriges_374222.distribuido_transacciones_bueno limit 10;

SELECT *
FROM prodriges_374222.distribuido_transacciones_bueno
LIMIT 10

Query id: 10033866-2b80-4e36-a4f4-bcd3e2d36987

┌─user_id_out─┬─user_id_in─┬─important─┬─amount─┬────────datetime─┐
│          15 │       4558 │         0 │ 491.17 │ 2018-01-13 09:18:21 │
│          15 │       5485 │         1 │ 247.88 │ 2018-01-07 06:08:06 │
│          15 │       2620 │         0 │ 340.09 │ 2018-01-03 08:18:50 │
│          15 │       6921 │         0 │ 222.92 │ 2018-01-19 10:13:36 │
│          15 │       9192 │         0 │ 516.45 │ 2018-01-31 20:13:41 │
│          15 │       2633 │         0 │ 622.75 │ 2018-01-10 19:37:42 │
│          15 │       7242 │         0 │ 953.76 │ 2018-01-04 17:17:26 │
│          15 │       9790 │         1 │ 760.72 │ 2018-01-27 22:45:59 │
│          15 │       5930 │         0 │ 225.31 │ 2018-01-15 20:57:08 │
│          15 │       5220 │         1 │ 355.24 │ 2018-01-19 22:21:28 │
└─────────────┴────────────┴───────────┴────────┴─────────────────────┘
```

3. Implement the chosen MVs. Also, you are able to create extra tables with different engines if you need them. The number of extra tables should be reasonable.

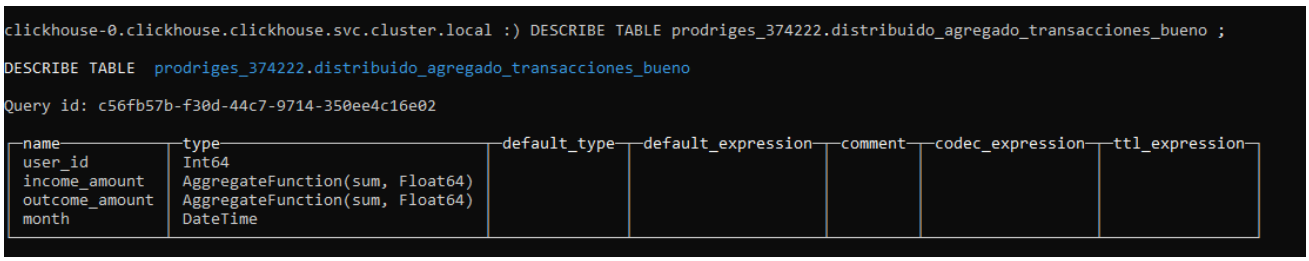
```
CREATE TABLE prodriges_374222.agregado_transacciones_bueno ON
CLUSTER kube_clickhouse_cluster (
    `user_id` Int64,
    `income_amount` AggregateFunction(sum, Float64),
    `outcome_amount` AggregateFunction(sum, Float64),
    `month` DateTime) ENGINE = AggregatingMergeTree
ORDER BY (user_id, month);
```

A table is created to store aggregated transaction data by user and month, using the AggregatingMergeTree engine and ordered by (user_id, month). This optimizes aggregation queries and facilitates monthly transaction analysis.

```
CREATE TABLE
prodriges_374222.distribuido_agregado_transacciones_bueno ON
CLUSTER kube_clickhouse_cluster AS
prodriges_374222.agregado_transacciones_bueno
ENGINE = Distributed(kube_clickhouse_cluster, prodriges_374222,
agregado_transacciones_bueno, xxHash64(user_id));
```

A distributed version of the previous table is created using the Distributed engine and xxHash64(user_id) as the shard key. This balances data distribution across the cluster, improving performance and availability.

```
DESCRIBE TABLE prodriges_374222.distribuido_agregado_transacciones_bueno ;
```



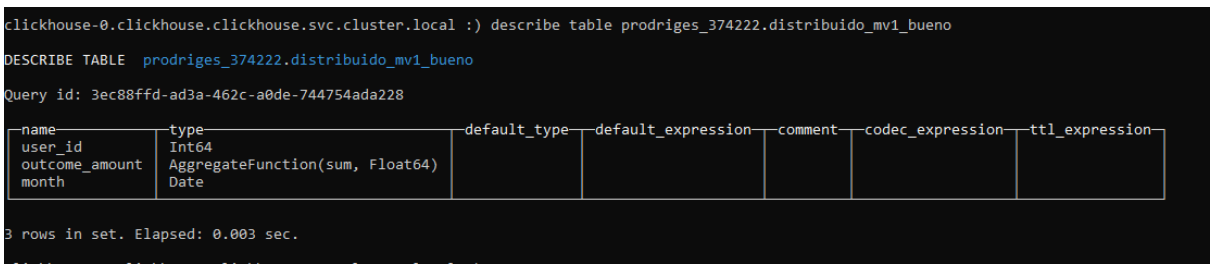

```
CREATE MATERIALIZED VIEW prodriges_374222.mv1_bueno ON CLUSTER
kube_clickhouse_cluster TO
prodriges_374222.distribuido_agregado_transacciones_bueno AS
SELECT
    user_id_out AS user_id,
    sumState(amount) AS outcome_amount,
    toDate(toStartOfMonth(datetime)) AS month
FROM prodriges_374222.transacciones_bueno
GROUP BY user_id, month;
```

A view is created to pre-aggregate expenses by user and month, directing the results to the distributed table. This preprocesses the data, enhancing the performance of subsequent queries.

```
CREATE TABLE prodriges_374222.distribuido_mv1_bueno ON CLUSTER
kube_clickhouse_cluster AS prodriges_374222.mv1_bueno
ENGINE = Distributed(kube_clickhouse_cluster, prodriges_374222,
mv1_bueno);
```

A distributed table based on the materialized view mv1_bueno is created to ensure quick and efficient access to pre-aggregated data.

```
Describe table prodriges_374222.distribuido_mv1_bueno;
```




```
CREATE MATERIALIZED VIEW prodriges_374222.mv2_bueno ON CLUSTER
kube_clickhouse_cluster TO
prodriges_374222.distribuido_agregado_transacciones_bueno AS
SELECT
    user_id_in AS user_id,
    sumState(amount) AS income_amount,
    toDate(toStartOfMonth(datetime)) AS month
```

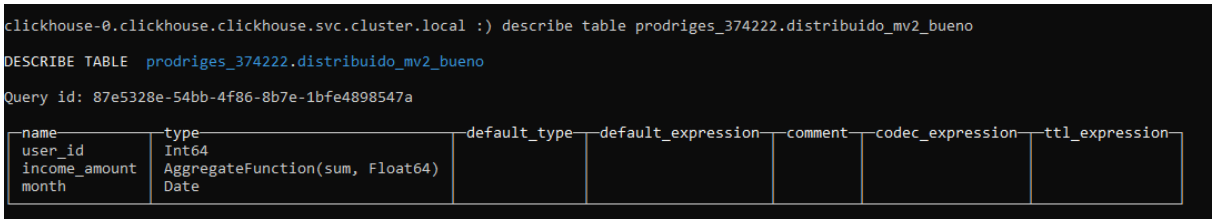
```
FROM prodriges_374222.transacciones_bueno
GROUP BY user_id, month;
```

Similar to mv1_bueno, but for incomes by user and month. This also preprocesses the data, optimizing income queries.

```
CREATE TABLE prodriges_374222.distribuido_mv2_bueno ON CLUSTER
kube_clickhouse_cluster AS prodriges_374222.mv2_bueno
ENGINE = Distributed(kube_clickhouse_cluster, prodriges_374222,
mv2_bueno);
```

A distributed table based on mv2_bueno is created, ensuring efficient access to pre-aggregated income data.

```
Describe table prodriges_374222.distribuido_mv2_bueno;
```



a. Users' saldo for the current moment.

```
CREATE MATERIALIZED VIEW prodriges_374222.mv3_bueno ON CLUSTER
kube_clickhouse_cluster
ENGINE = SummingMergeTree
ORDER BY user_id AS
SELECT
    user_id AS user_id,
    sumMerge(income_amount) - sumMerge(outcome_amount) AS
current_balance
FROM prodriges_374222.agregado_transacciones_bueno
GROUP BY user_id;
```

The materialized view prodriges_374222.mv3_bueno is created to optimize user balance calculations. Using the SummingMergeTree engine, it orders data by user_id and pre-aggregates balances by subtracting aggregated expenses from aggregated incomes, grouped by user_id. This pre-aggregation improves the performance of balance queries.

```
CREATE TABLE prodriges_374222.distribuido_mv3_bueno ON CLUSTER
kube_clickhouse_cluster AS prodriges_374222.mv3_bueno
ENGINE = Distributed(kube_clickhouse_cluster, prodriges_374222,
mv3_bueno);
```

The table prodriges_374222.distribuido_mv3_bueno is a distributed version of mv3_bueno, using the Distributed engine. This ensures balanced data distribution across the kube_clickhouse_cluster, enhancing query performance and availability. Data is evenly distributed using xxHash64(user_id) to prevent bottlenecks.

```
Describe table prodriges_374222.distribuido_mv3_bueno;
```

```
clickhouse-0.clickhouse.clickhouse.svc.cluster.local :) describe table prodriges_374222.distribuido_mv3_bueno

DESCRIBE TABLE prodriges_374222.distribuido_mv3_bueno

Query id: 3b833719-ec1b-45f6-a0f4-669a2be40e88

┌─name──────────┬─type────┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ user_id       │ Int64   │              │                    │         │                  │                │
│ current_balance │ Float64 │              │                    │         │                  │                │
└───────────────┴─────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

```
SELECT * FROM prodriges_374222.distribuido_mv3_bueno LIMIT 20;
```

```
clickhouse-0.clickhouse.clickhouse.svc.cluster.local :) select * from prodriges_374222.distribuido_mv3_bueno limit 20;

SELECT *
FROM prodriges_374222.distribuido_mv3_bueno
LIMIT 20

Query id: 100336a7-083f-45ba-a147-4bab36d059bc

┌─user_id─┬──current_balance─┐
│      15 │  26815.23000000007 │
│      27 │ -25708.169999999875 │
│      28 │   7568.790000000081 │
│      31 │  -4068.820000000065 │
│      40 │ -1477.7100000001374 │
│      44 │  -8039.220000000118 │
│      65 │ -21188.450000000084 │
│     133 │   4554.360000000015 │
│     150 │ -17313.609999999848 │
│     152 │   52146.11999999982 │
│     155 │  10108.180000000095 │
│     164 │ -28045.889999999934 │
│     202 │  -9632.409999999989 │
│     204 │  -4798.500000000073 │
│     215 │  -7466.370000000054 │
│     224 │ -15761.409999999989 │
│     240 │   36187.79999999994 │
│     244 │ -17187.390000000072 │
│     260 │  -30468.60000000008 │
│     265 │  21159.790000000095 │
└─────────┴──────────────────┘
```

b. The sums for incoming and outcoming transactions by months for each user.

```
CREATE MATERIALIZED VIEW prodriges_374222.mv4_bueno ON CLUSTER
kube_clickhouse_cluster
ENGINE = SummingMergeTree
ORDER BY (user_id, month) AS
SELECT
    user_id AS user_id,
    sumMerge(income_amount) AS total_income,
    sumMerge(outcome_amount) AS total_outcome,
    formatDateTime(month, '%m.%Y') AS month
FROM prodriges_374222. agregado_transacciones_bueno
GROUP BY user_id, month;
```

The view prodriges_374222.mv4_bueno aggregates monthly income and outcome for each user. Using the SummingMergeTree engine, it orders by (user_id, month) and pre-aggregates data by summing income_amount and outcome_amount, formatted by month and year. This setup enhances query performance for monthly user transactions.

```
CREATE TABLE prodriges_374222.distribuido_mv4_bueno ON CLUSTER
kube_clickhouse_cluster AS prodriges_374222.mv4_bueno
ENGINE = Distributed(kube_clickhouse_cluster, prodriges_374222,
mv4_bueno);
```

The table prodriges_374222.distribuido_mv4_bueno is a distributed version of mv4_bueno, using the Distributed engine. It ensures balanced data distribution across the kube_clickhouse_cluster, improving performance and availability by evenly spreading the load.

```
Describe table prodriges_374222.distribuido_mv4_bueno;
```

```
clickhouse-0.clickhouse.clickhouse.svc.cluster.local :) Describe table prodriges_374222.distribuido_mv4_bueno;

DESCRIBE TABLE  prodriges_374222.distribuido_mv4_bueno

Query id: ebd3d60e-b787-4866-8eba-2ae31f0747ee

┌─name──────────┬─type────┬─default_type─┬─default_expression─┬─comment─┬─codec_expression─┬─ttl_expression─┐
│ user_id       │ Int64   │              │                    │         │                  │                │
│ total_income  │ Float64 │              │                    │         │                  │                │
│ total_outcome │ Float64 │              │                    │         │                  │                │
│ month         │ String  │              │                    │         │                  │                │
└───────────────┴─────────┴──────────────┴────────────────────┴─────────┴──────────────────┴────────────────┘
```

```
SELECT * FROM prodriges_374222.distribuido_mv4_bueno LIMIT 20;
```

```
clickhouse-0.clickhouse.clickhouse.svc.cluster.local :) select * from prodriges_374222.distribuido_mv4_bueno limit 20;

SELECT *
FROM prodriges_374222.distribuido_mv4_bueno
LIMIT 20

Query id: da1721c6-4ea8-46c2-be19-fd47343a0e1f

┌─user_id─┬──────total_income─┬─────total_outcome─┬─month─┐
│      15 │          53468.69 │  42730.83999999999 │ 01.2018 │
│      15 │ 49846.920000000006 │  42387.64000000001 │ 02.2018 │
│      15 │          50819.01 │  38248.51000000002 │ 03.2018 │
│      15 │          51008.34 │           58160.13 │ 04.2018 │
│      15 │  51864.34000000001 │  58426.03000000002 │ 05.2018 │
│      15 │          54987.63 │ 57899.450000000004 │ 06.2018 │
│      15 │  61306.11999999999 │  54940.44999999997 │ 07.2018 │
│      15 │  49134.20999999999 │ 53875.699999999975 │ 08.2018 │
│      15 │          45626.81 │  47324.35000000002 │ 09.2018 │
│      15 │          57219.08 │ 46037.520000000004 │ 10.2018 │
│      15 │          46351.06 │  46339.17000000001 │ 11.2018 │
│      15 │ 50137.479999999996 │ 48584.670000000006 │ 12.2018 │
│      27 │ 45806.799999999996 │  56446.17000000001 │ 01.2018 │
│      27 │          39580.17 │           40475.49 │ 02.2018 │
│      27 │  58813.98999999999 │ 49819.130000000005 │ 03.2018 │
│      27 │          58081.66 │           47982.55 │ 04.2018 │
│      27 │  41411.00000000001 │           55516.17 │ 05.2018 │
│      27 │ 53572.189999999995 │  54257.39999999998 │ 06.2018 │
│      27 │  43223.24999999999 │  56839.96999999998 │ 07.2018 │
│      27 │          47609.14 │  47214.86999999998 │ 08.2018 │
└─────────┴───────────────────┴────────────────────┴─────────┘
```