Optimizing Time-Based Segment Evaluation with ClickHouse.

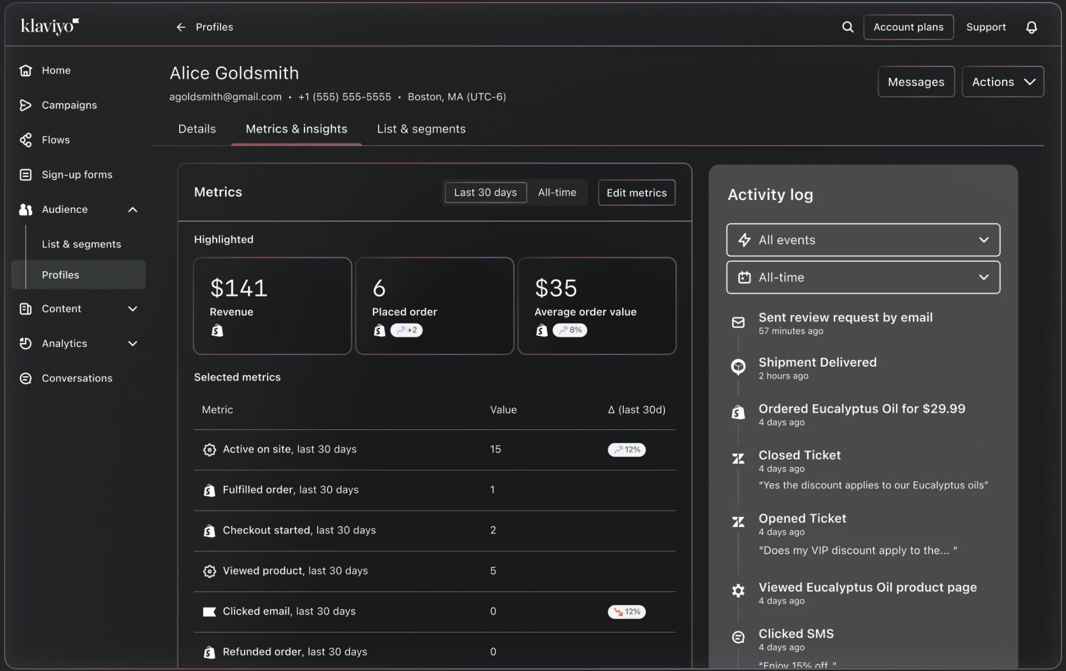
klaviyo

An overview of segmentation.

01

An overview of segmentation.

User Profiles



klaviyo

← Profiles

Account plans · Support

- Home
- Campaigns
- Flows
- Sign-up forms
- Audience
  - List & segments
  - Profiles
- Content
- Analytics
- Conversations

Alice Goldsmith

agoldsmith@gmail.com • +1 (555) 555-5555 • Boston, MA (UTC-6)

Messages · Actions

Details · Metrics & insights · List & segments

Metrics

Last 30 days · All-time · Edit metrics

Highlighted

$141 Revenue	6 Placed order +2	$35 Average order value 8%

Selected metrics

Metric	Value	Δ (last 30d)
Active on site, last 30 days	15	↗12%
Fulfilled order, last 30 days	1	
Checkout started, last 30 days	2	
Viewed product, last 30 days	5	
Clicked email, last 30 days	0	↘12%
Refunded order, last 30 days	0	

Activity log

All events

All-time

Sent review request by email
57 minutes ago

Shipment Delivered
2 hours ago

Ordered Eucalyptus Oil for $29.99
4 days ago

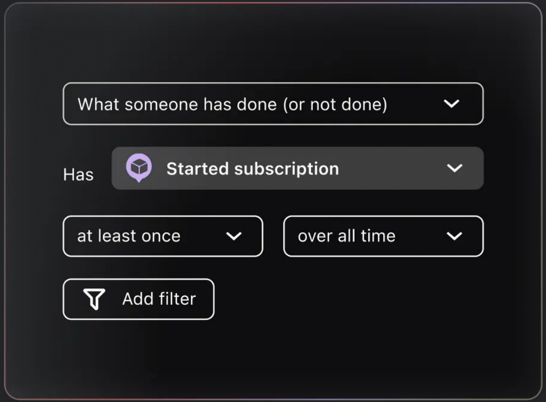
Closed Ticket
4 days ago
"Yes the discount applies to our Eucalyptus oils"

Opened Ticket
4 days ago
"Does my VIP discount apply to the... "

Viewed Eucalyptus Oil product page
4 days ago

Clicked SMS
4 days ago
"Enjoy 15% off."

Segments



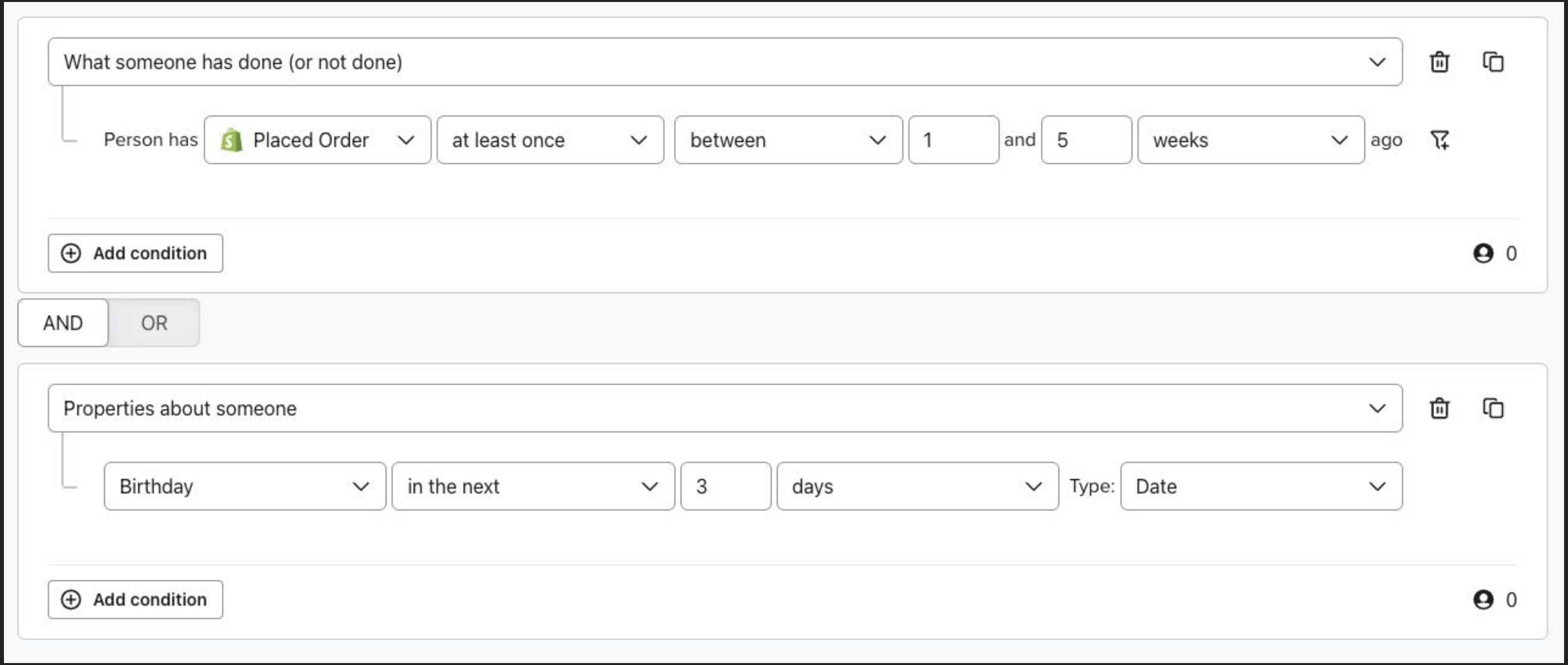
What someone has done (or not done)

Has Started subscription

at least once · over all time

Add filter

An overview of segmentation.


What someone has done (or not done)

Person has Placed Order at least once between 1 and 5 weeks ago

Add condition 0

AND OR

Properties about someone

Birthday in the next 3 days Type: Date

Add condition 0

01

Initial Segment Creation

Initial population of a segment. When creating or updating a segment.

02

Event-Driven Real Time Updates

Update segment memberships for user profiles as we receive new data.

03

Time-Based Periodic Updates

Update segment memberships for segments with relative time conditions.

ClickHouse Cluster

- 192 hosts
- Bi-Level sharding
- 8 layers, each with 8 shards
- 3 replicas per shard

segment_defs_local

Segment definition data.

events_local

A time-series log of events, partitioned by month.

traits_local

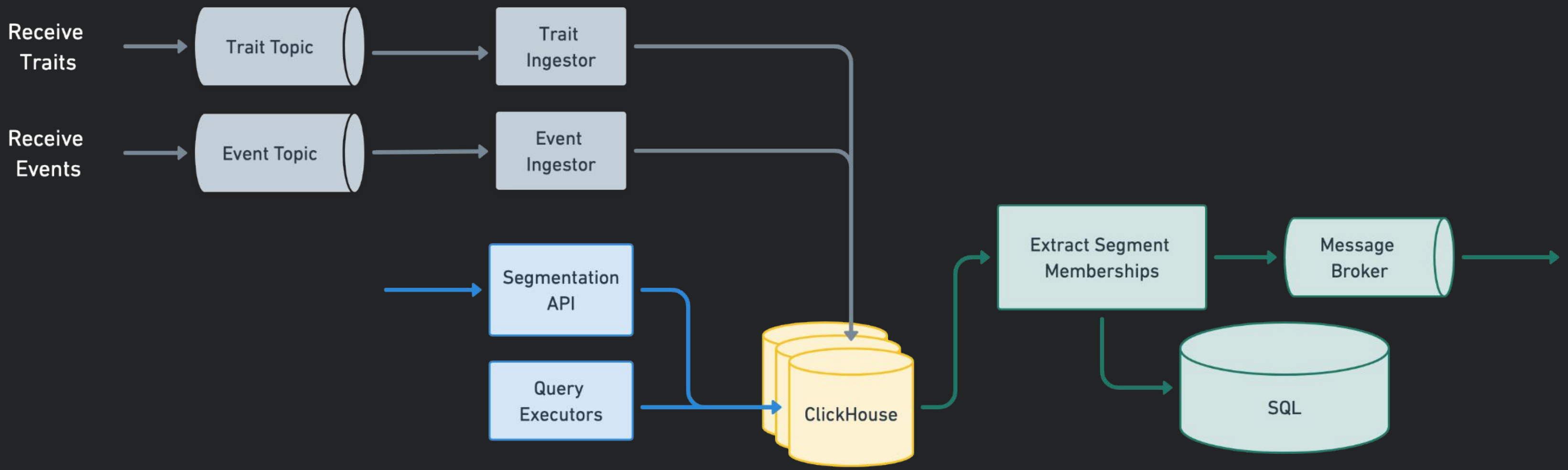
Stores snapshots of profile data.

Replacing merge tree, insert new record when data changes.

segment_results_log_local

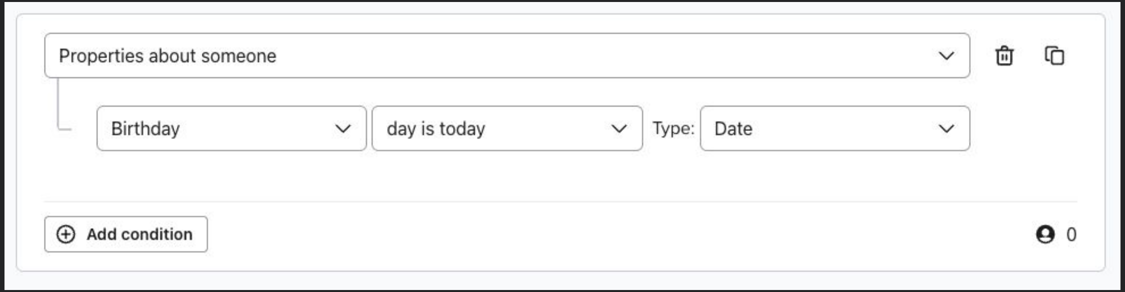
Queries write segment membership results here.

An overview of segmentation.



Time Based
Periodic Updates

02



Properties about someone

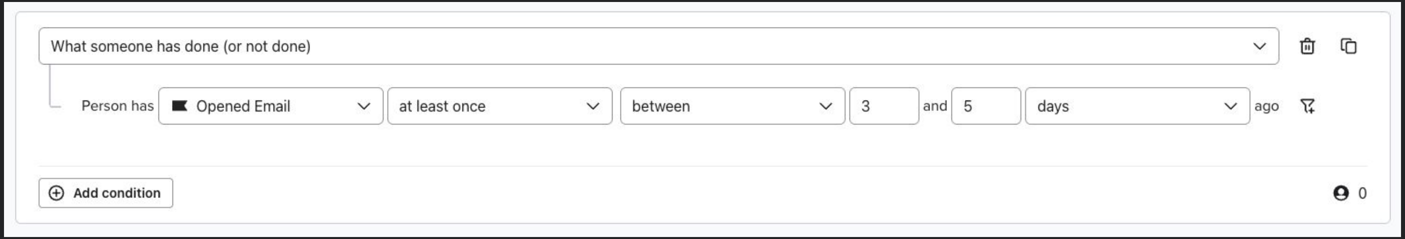
Birthday | day is today | Type: Date

“Birthday” property is today

23rd	24th	25th (Today)
		🟩

23rd	24th	25th	26th (Today)
			🟩

23rd	24th	25th	26th	27th (Today)
				🟩


What someone has done (or not done)

Person has Opened Email at least once between 3 and 5 days ago

Between 3 and 5 days ago

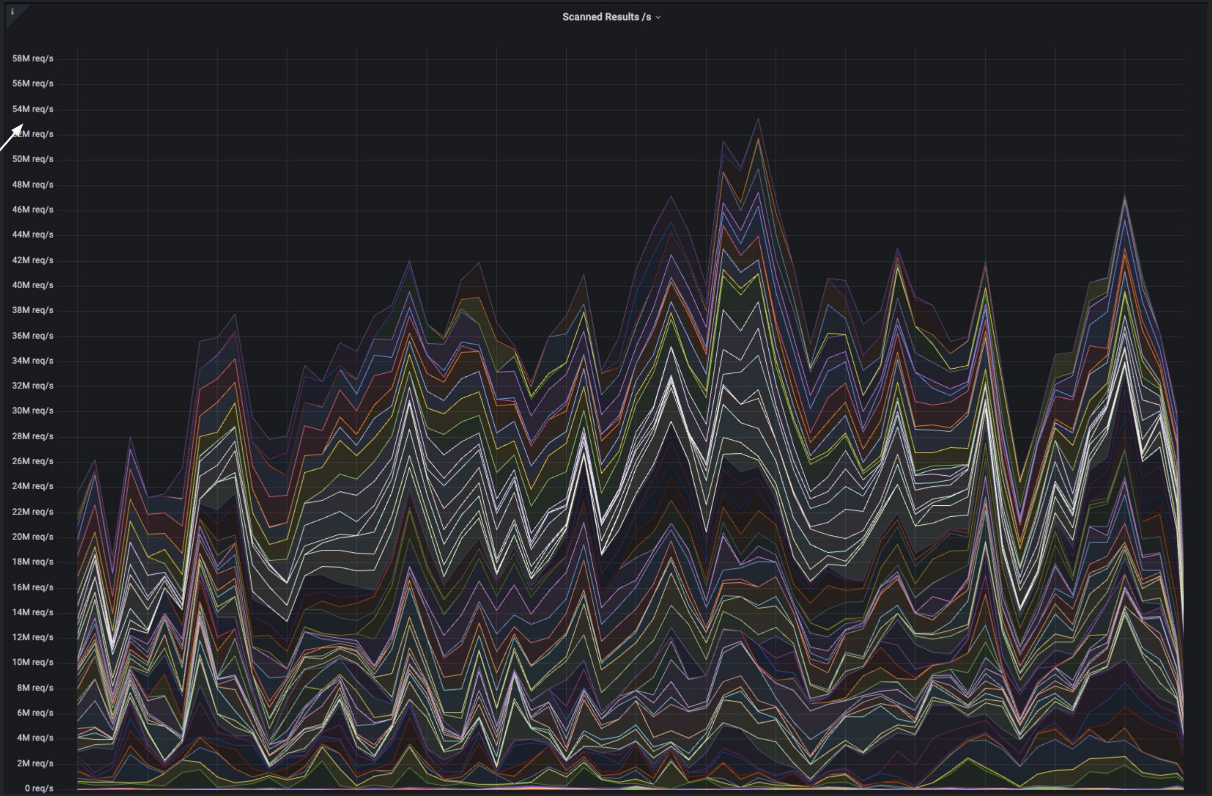
17th	18th	19th	20th	21st	22nd	23rd	24th	25th (Today)
			🟩	🟩	🟩			

17th	18th	19th	20th	21st	22nd	23rd	24th	25th	26th (Today)
				🟩	🟩	🟩			

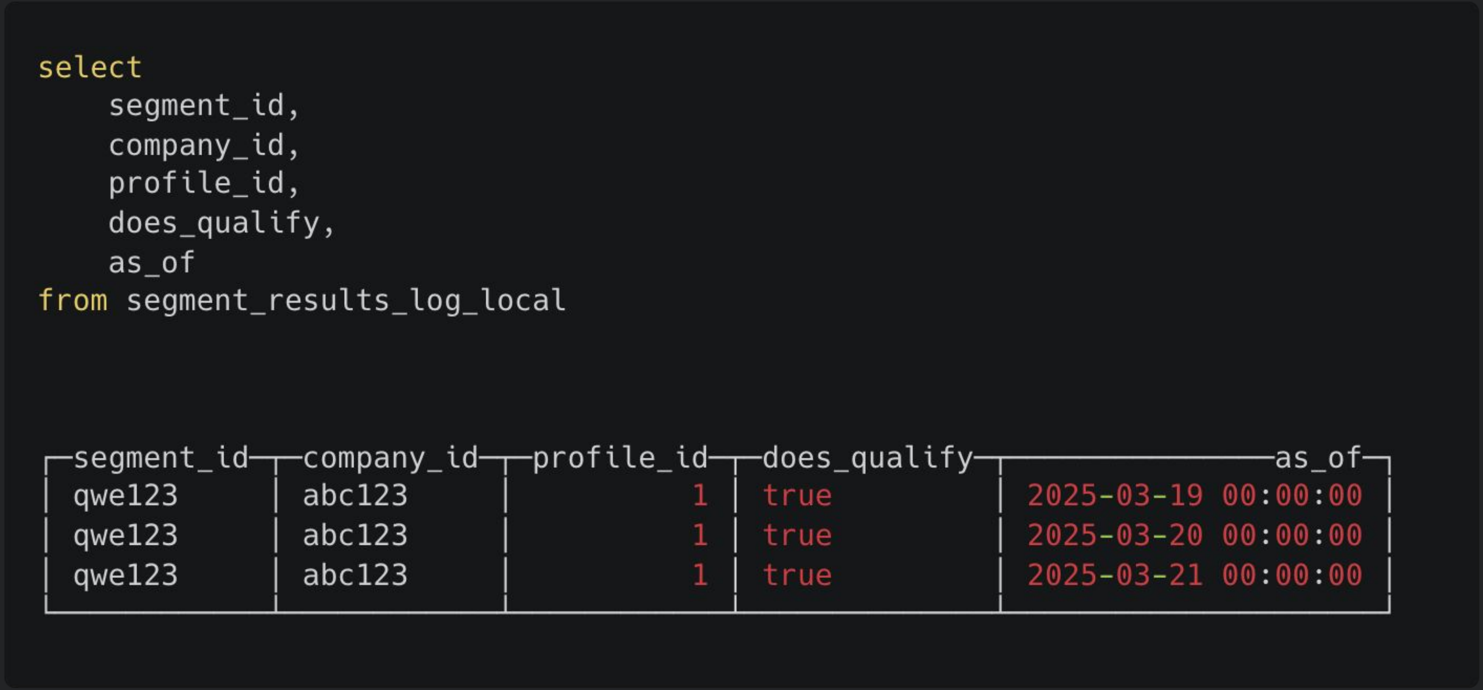
17th	18th	19th	20th	21st	22nd	23rd	24th	25th	26th	27th (Today)
					🟩	🟩	🟩			

Time based periodic updates.

53 million req/s!



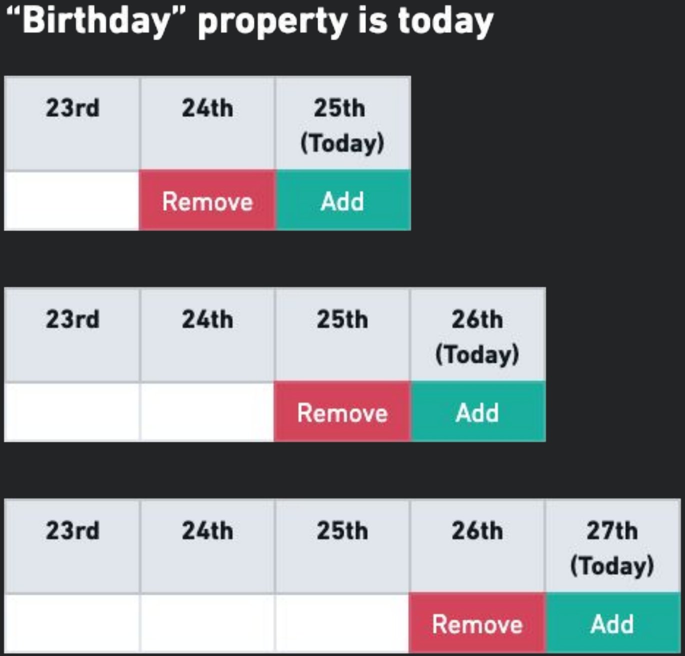
Time based periodic updates.



```
select
    segment_id,
    company_id,
    profile_id,
    does_qualify,
    as_of
from segment_results_log_local
```

```
┌─segment_id─┬─company_id─┬─profile_id─┬─does_qualify─┬───────────────as_of─┐
│ qwe123     │ abc123     │          1 │ true         │ 2025-03-19 00:00:00 │
│ qwe123     │ abc123     │          1 │ true         │ 2025-03-20 00:00:00 │
│ qwe123     │ abc123     │          1 │ true         │ 2025-03-21 00:00:00 │
└────────────┴────────────┴────────────┴──────────────┴─────────────────────┘
```

Time based periodic updates.



“Birthday” property is today

23rd	24th	25th (Today)
	Remove	Add

23rd	24th	25th	26th (Today)
		Remove	Add

23rd	24th	25th	26th	27th (Today)
			Remove	Add

Time based periodic updates.

Between 3 and 5 days ago

17th	18th	19th	20th	21st	22nd	23rd	24th	25th (Today)
		Remove	▨	▨	Add			

17th	18th	19th	20th	21st	22nd	23rd	24th	25th	26th (Today)
			Remove	▨	▨	Add			

17th	18th	19th	20th	21st	22nd	23rd	24th	25th	26th	27th (Today)
				Remove	▨	▨	Add			

Instead of updating segment memberships for *every* profile…

Run a lightweight query against the event and trait tables, to identify profiles with events or properties within specific time bounds.

Only update segment memberships for *these* profiles.

Time based periodic updates.

events_local

Time Series Aligned ✅

Since events are ordered by timestamp, we can easily identify profiles with events during specified time bounds.

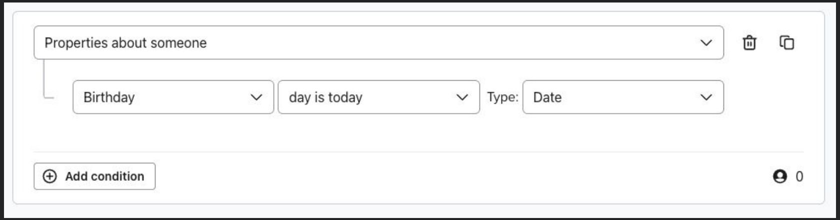
traits_local

Not Ordered By Timestamp ❌

Traits are snapshots of profile data, and thus aren’t ordered by timestamp.

Read every trait row to identify impacted profiles 👎.

Time based periodic updates.


Properties about someone

Birthday | day is today | Type: Date

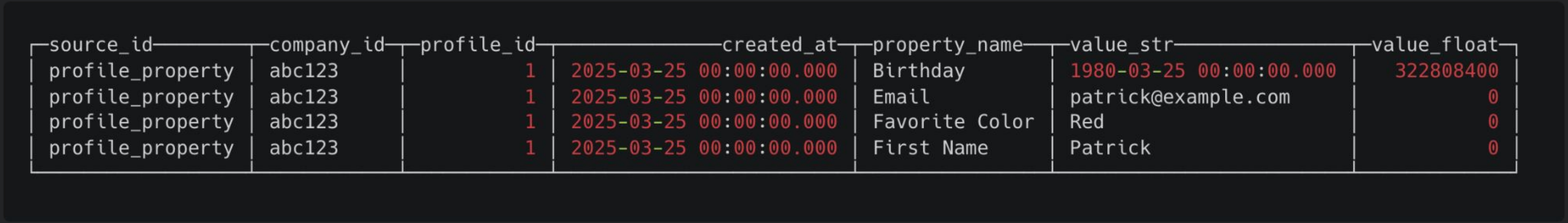
```
source_id:              profile_property
company_id:             abc123
profile_id:             1
created_at:             2025-03-25 00:00:00.000
property_keys:          ['Email',                'First Name',   'Favorite Color',   'Birthday'                 ]
property_vals_str:      ['patrick@example.com',  'Patrick',      'Red',              '1980-03-25 00:00:00.000']
property_vals_numeric:  [0,                      0,              0,                  322808400                  ]
```

Time based periodic updates.

```sql
create table traits_eav_local
(
    source_id LowCardinality(String),

    company_id String,
    profile_id UInt32 CODEC(Delta(4), LZ4),
    created_at DateTime64(3) CODEC(T64, LZ4),

    property_name String,
    value_str String,
    value_float Float64
)
ENGINE = ReplicatedReplacingMergeTree(created_at)
PARTITION BY source_id
ORDER BY (company_id, property_name, profile_id)
SETTINGS index_granularity = ...
```

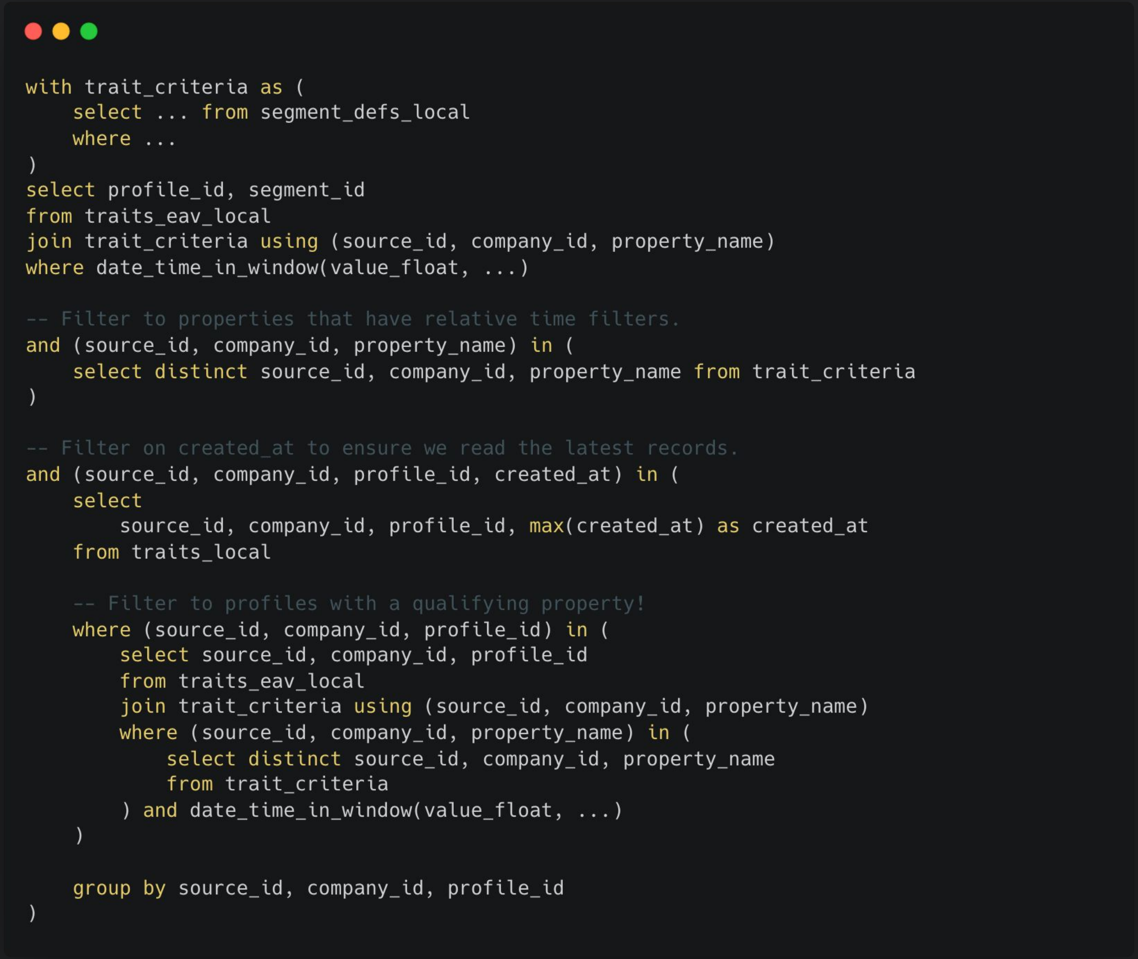
```sql
CREATE MATERIALIZED VIEW mv_traits_to_traits_eav
TO traits_eav_local
AS
SELECT
    source_id,

    company_id,
    profile_id,
    created_at,

    property_name,
    value_str,
    value_float
FROM traits_local
ARRAY JOIN
    property_keys as property_name,
    property_vals_str as value_str,
    property_vals_numeric as value_float
```

Time based periodic updates.



```
┌─source_id────────┬─company_id─┬─profile_id─┬──────────────created_at─┬─property_name──┬─value_str─────────────────┬─value_float─┐
│ profile_property │ abc123     │          1 │ 2025-03-25 00:00:00.000 │ Birthday       │ 1980-03-25 00:00:00.000   │   322808400 │
│ profile_property │ abc123     │          1 │ 2025-03-25 00:00:00.000 │ Email          │ patrick@example.com       │           0 │
│ profile_property │ abc123     │          1 │ 2025-03-25 00:00:00.000 │ Favorite Color │ Red                       │           0 │
│ profile_property │ abc123     │          1 │ 2025-03-25 00:00:00.000 │ First Name     │ Patrick                   │           0 │
└──────────────────┴────────────┴────────────┴─────────────────────────┴────────────────┴───────────────────────────┴─────────────┘
```

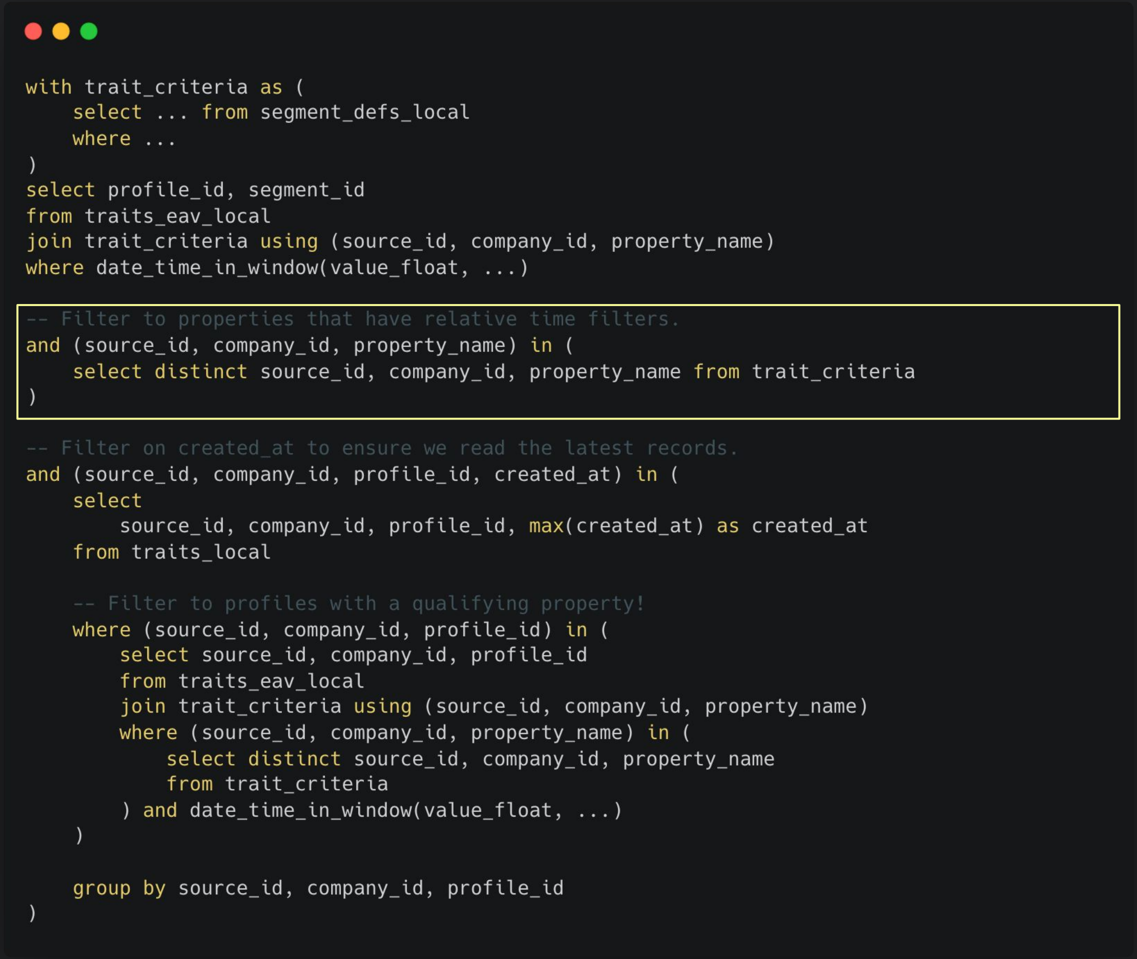
What `(profile_id, segment_id)` pairs need to be updated?



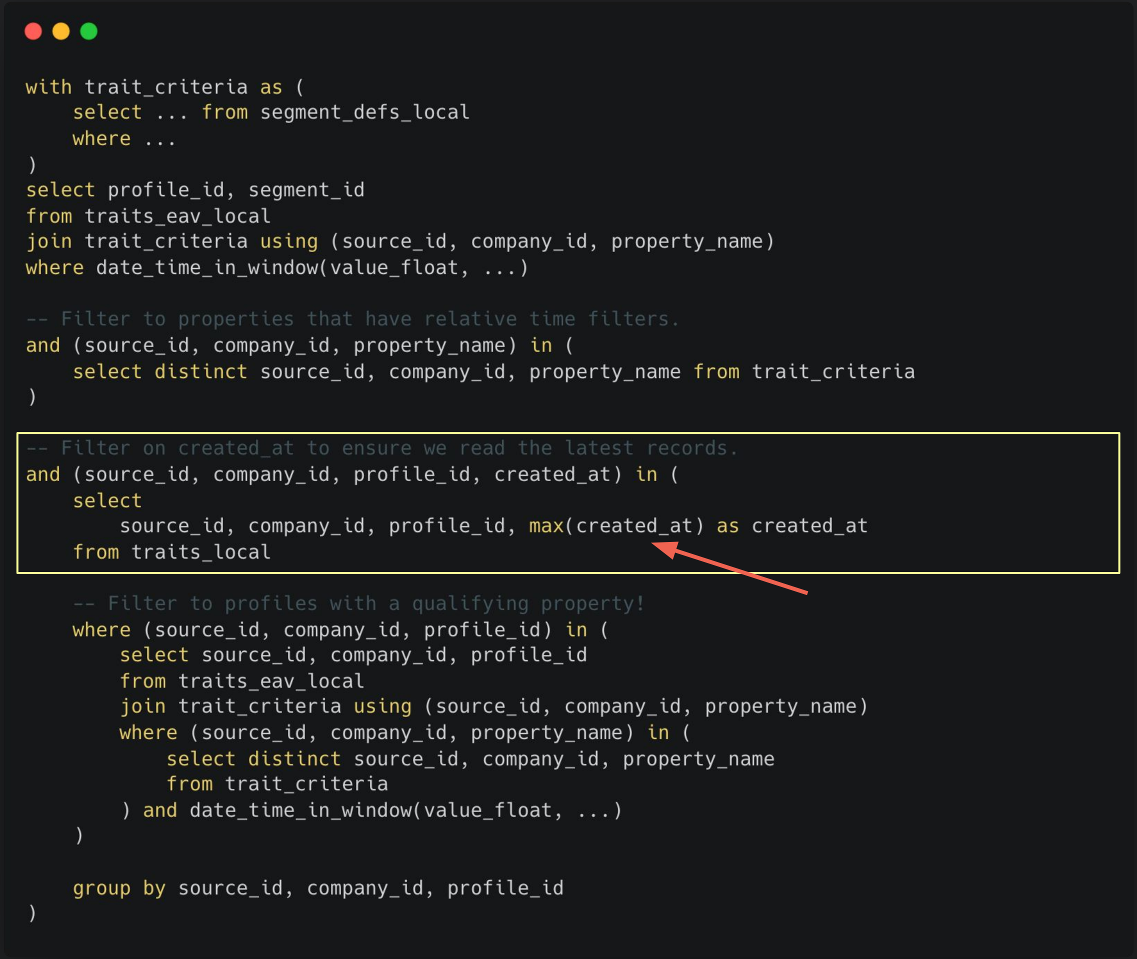
```sql
with trait_criteria as (
    select ... from segment_defs_local
    where ...
)
select profile_id, segment_id
from traits_eav_local
join trait_criteria using (source_id, company_id, property_name)
where date_time_in_window(value_float, ...)

-- Filter to properties that have relative time filters.
and (source_id, company_id, property_name) in (
    select distinct source_id, company_id, property_name from trait_criteria
)

-- Filter on created_at to ensure we read the latest records.
and (source_id, company_id, profile_id, created_at) in (
    select
        source_id, company_id, profile_id, max(created_at) as created_at
    from traits_local

    -- Filter to profiles with a qualifying property!
    where (source_id, company_id, profile_id) in (
        select source_id, company_id, profile_id
        from traits_eav_local
        join trait_criteria using (source_id, company_id, property_name)
        where (source_id, company_id, property_name) in (
            select distinct source_id, company_id, property_name
            from trait_criteria
        ) and date_time_in_window(value_float, ...)
    )

    group by source_id, company_id, profile_id
)
```

```sql
with trait_criteria as (
    select ... from segment_defs_local
    where ...
)
select profile_id, segment_id
from traits_eav_local
join trait_criteria using (source_id, company_id, property_name)
where date_time_in_window(value_float, ...)

-- Filter to properties that have relative time filters.
and (source_id, company_id, property_name) in (
    select distinct source_id, company_id, property_name from trait_criteria
)

-- Filter on created_at to ensure we read the latest records.
and (source_id, company_id, profile_id, created_at) in (
    select
        source_id, company_id, profile_id, max(created_at) as created_at
    from traits_local

    -- Filter to profiles with a qualifying property!
    where (source_id, company_id, profile_id) in (
        select source_id, company_id, profile_id
        from traits_eav_local
        join trait_criteria using (source_id, company_id, property_name)
        where (source_id, company_id, property_name) in (
            select distinct source_id, company_id, property_name
            from trait_criteria
        ) and date_time_in_window(value_float, ...)
    )

    group by source_id, company_id, profile_id
)
```

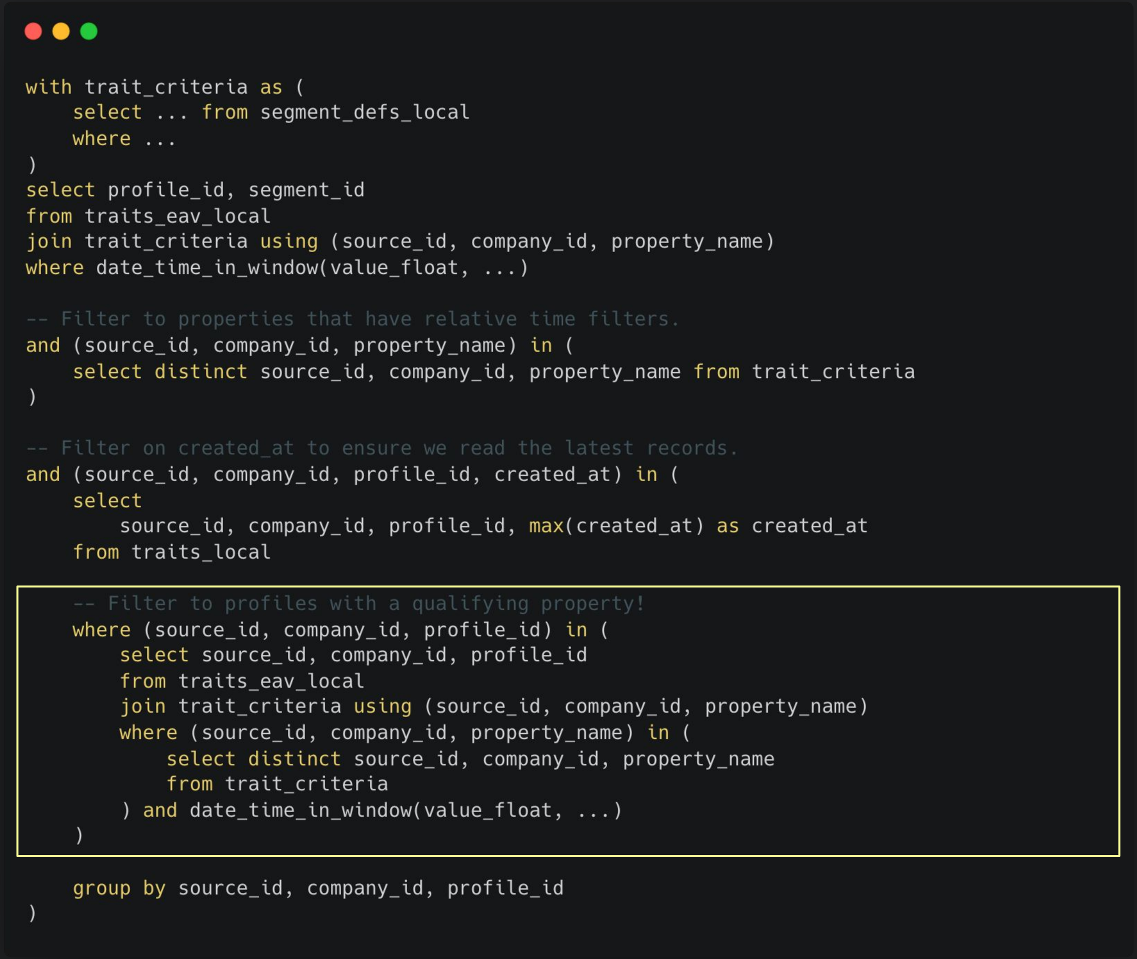
```sql
with trait_criteria as (
    select ... from segment_defs_local
    where ...
)
select profile_id, segment_id
from traits_eav_local
join trait_criteria using (source_id, company_id, property_name)
where date_time_in_window(value_float, ...)

-- Filter to properties that have relative time filters.
and (source_id, company_id, property_name) in (
    select distinct source_id, company_id, property_name from trait_criteria
)

-- Filter on created_at to ensure we read the latest records.
and (source_id, company_id, profile_id, created_at) in (
    select
        source_id, company_id, profile_id, max(created_at) as created_at
    from traits_local

    -- Filter to profiles with a qualifying property!
    where (source_id, company_id, profile_id) in (
        select source_id, company_id, profile_id
        from traits_eav_local
        join trait_criteria using (source_id, company_id, property_name)
        where (source_id, company_id, property_name) in (
            select distinct source_id, company_id, property_name
            from trait_criteria
        ) and date_time_in_window(value_float, ...)
    )

    group by source_id, company_id, profile_id
)
```



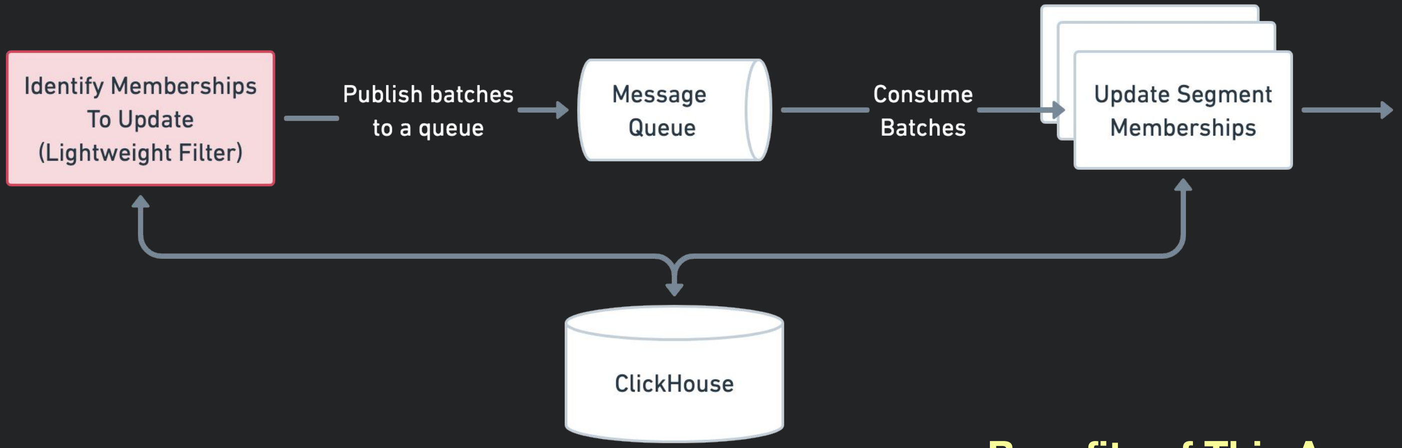
```sql
with trait_criteria as (
    select ... from segment_defs_local
    where ...
)
select profile_id, segment_id
from traits_eav_local
join trait_criteria using (source_id, company_id, property_name)
where date_time_in_window(value_float, ...)

-- Filter to properties that have relative time filters.
and (source_id, company_id, property_name) in (
    select distinct source_id, company_id, property_name from trait_criteria
)

-- Filter on created_at to ensure we read the latest records.
and (source_id, company_id, profile_id, created_at) in (
    select
        source_id, company_id, profile_id, max(created_at) as created_at
    from traits_local

    -- Filter to profiles with a qualifying property!
    where (source_id, company_id, profile_id) in (
        select source_id, company_id, profile_id
        from traits_eav_local
        join trait_criteria using (source_id, company_id, property_name)
        where (source_id, company_id, property_name) in (
            select distinct source_id, company_id, property_name
            from trait_criteria
        ) and date_time_in_window(value_float, ...)
    )

    group by source_id, company_id, profile_id
)
```



```sql
with trait_criteria as (
    select ... from segment_defs_local
    where ...
)
select profile_id, segment_id
from traits_eav_local
join trait_criteria using (source_id, company_id, property_name)
where date_time_in_window(value_float, ...)

-- Filter to properties that have relative time filters.
and (source_id, company_id, property_name) in (
    select distinct source_id, company_id, property_name from trait_criteria
)

-- Filter on created_at to ensure we read the latest records.
and (source_id, company_id, profile_id, created_at) in (
    select
        source_id, company_id, profile_id, max(created_at) as created_at
    from traits_local

    -- Filter to profiles with a qualifying property!
    where (source_id, company_id, profile_id) in (
        select source_id, company_id, profile_id
        from traits_eav_local
        join trait_criteria using (source_id, company_id, property_name)
        where (source_id, company_id, property_name) in (
            select distinct source_id, company_id, property_name
            from trait_criteria
        ) and date_time_in_window(value_float, ...)
    )

    group by source_id, company_id, profile_id
)
```



```sql
with trait_criteria as (
    select ... from segment_defs_local
    where ...
)
select profile_id, segment_id
from traits_eav_local
join trait_criteria using (source_id, company_id, property_name)
where date_time_in_window(value_float, ...)

-- Filter to properties that have relative time filters.
and (source_id, company_id, property_name) in (
    select distinct source_id, company_id, property_name from trait_criteria
)

-- Filter on created_at to ensure we read the latest records.
and (source_id, company_id, profile_id, created_at) in (
    select
        source_id, company_id, profile_id, max(created_at) as created_at
    from traits_local

    -- Filter to profiles with a qualifying property!
    where (source_id, company_id, profile_id) in (
        select source_id, company_id, profile_id
        from traits_eav_local
        join trait_criteria using (source_id, company_id, property_name)
        where (source_id, company_id, property_name) in (
            select distinct source_id, company_id, property_name
            from trait_criteria
        ) and date_time_in_window(value_float, ...)
    )

    group by source_id, company_id, profile_id
)
```


Identify Memberships To Update (Lightweight Filter)
Publish batches to a queue
Message Queue
Consume Batches
Update Segment Memberships
ClickHouse

Benefits of This Approach

Enables easy horizontal-scaling.

The best-effort filtering step is fast & lightweight. Updating segment memberships is more compute heavy, so parallelize this.

Josh Bradt

Vinay Garg

Isabel Yap

Mike Burton

Jeff Johansen

Himnish Hunma

Sarah Fida

Kanav Bengani

Andrew Volkmann

Kavya Sri Thadakamalla

klaviyo

careers.klaviyo.com