WITH

-- Step 1: Precompute the creator for each message

message\_creators AS (

SELECT

m.m\_messageid,

m.m\_creatorid

FROM

message m

),

-- Step 2: Precompute foreign likes for each message

foreign\_likes AS (

SELECT

l.l\_messageid AS message\_id,

COUNT(l.l\_personid) AS foreign\_like\_count

FROM

likes l

LEFT JOIN knows k ON l.l\_personid = k.k\_person2id -- Check if the liker is in the "knows" table

LEFT JOIN message\_creators mc ON k.k\_person1id = mc.m\_creatorid

WHERE

mc.m\_creatorid IS NULL -- Exclude likes from friends of the creator

GROUP BY

l.l\_messageid

),

-- Step 3: Precompute total likes for each message

total\_likes AS (

SELECT

l\_messageid AS message\_id,

COUNT(l\_personid) AS total\_like\_count

FROM

likes

GROUP BY

l\_messageid

)

SELECT

m.m\_messageid AS message\_id, -- The ID of the message

tl.total\_like\_count, -- Total number of likes for the message

fl.foreign\_like\_count, -- Number of foreign likes for the message

COALESCE(fl.foreign\_like\_count \* 1.0 / NULLIF(tl.total\_like\_count, 0), 0) AS foreign\_like\_ratio

-- Ratio of foreign likes to total likes

FROM

message m

JOIN

total\_likes tl ON m.m\_messageid = tl.message\_id

LEFT JOIN

foreign\_likes fl ON m.m\_messageid = fl.message\_id

WHERE

m.m\_length > 100 -- Filter messages longer than 100 characters

AND tl.total\_like\_count >= 20 -- Filter messages with at least 20 total likes

AND COALESCE(fl.foreign\_like\_count, 0) >= tl.total\_like\_count / 2

-- Ensure at least half of the likes are foreign

ORDER BY

foreign\_like\_ratio DESC; -- Sort by the foreign-like ratio in descending order

I want you to extend this query to include the full name of the first foreign person who liked the message and the time when they created the like. The attached schema file might be helpful to you.

Chat GPT Response ….

###### **ChatGPT said:**

ChatGPT

The new schema file appears to have the same structure as the previous one. Since the relevant details were identified earlier (e.g., likes, person, and knows tables), I will directly extend the query to include the full name of the first foreign liker and the time when the like was created. Here's the extended query:

sql

Copy code

WITH

-- Step 1: Precompute the creator for each message

message\_creators AS (

SELECT

m.m\_messageid,

m.m\_creatorid

FROM

message m

),

-- Step 2: Precompute foreign likes for each message

foreign\_likes AS (

SELECT

l.l\_messageid AS message\_id,

COUNT(l.l\_personid) AS foreign\_like\_count

FROM

likes l

LEFT JOIN knows k ON l.l\_personid = k.k\_person2id -- Check if the liker is in the "knows" table

LEFT JOIN message\_creators mc ON k.k\_person1id = mc.m\_creatorid

WHERE

mc.m\_creatorid IS NULL -- Exclude likes from friends of the creator

GROUP BY

l.l\_messageid

),

-- Step 3: Precompute total likes for each message

total\_likes AS (

SELECT

l\_messageid AS message\_id,

COUNT(l\_personid) AS total\_like\_count

FROM

likes

GROUP BY

l\_messageid

),

-- Step 4: Get the first foreign liker for each message

first\_foreign\_likers AS (

SELECT DISTINCT ON (l.l\_messageid)

l.l\_messageid AS message\_id,

l.l\_personid AS person\_id,

l.l\_creationdate AS like\_creation\_date

FROM

likes l

LEFT JOIN knows k ON l.l\_personid = k.k\_person2id

LEFT JOIN message\_creators mc ON k.k\_person1id = mc.m\_creatorid

WHERE

mc.m\_creatorid IS NULL -- Exclude likes from friends of the creator

ORDER BY

l.l\_messageid, l.l\_creationdate ASC -- Get the earliest like for each message

)

SELECT

m.m\_messageid AS message\_id, -- The ID of the message

tl.total\_like\_count, -- Total number of likes for the message

fl.foreign\_like\_count, -- Number of foreign likes for the message

COALESCE(fl.foreign\_like\_count \* 1.0 / NULLIF(tl.total\_like\_count, 0), 0) AS foreign\_like\_ratio,

-- Ratio of foreign likes to total likes

CONCAT(p.p\_firstname, ' ', p.p\_lastname) AS first\_foreign\_liker\_name, -- Full name of the first foreign liker

ffl.like\_creation\_date -- Time when the first foreign like was created

FROM

message m

JOIN

total\_likes tl ON m.m\_messageid = tl.message\_id

LEFT JOIN

foreign\_likes fl ON m.m\_messageid = fl.message\_id

LEFT JOIN

first\_foreign\_likers ffl ON m.m\_messageid = ffl.message\_id

LEFT JOIN

person p ON ffl.person\_id = p.p\_personid

WHERE

m.m\_length > 100 -- Filter messages longer than 100 characters

AND tl.total\_like\_count >= 20 -- Filter messages with at least 20 total likes

AND COALESCE(fl.foreign\_like\_count, 0) >= tl.total\_like\_count / 2

-- Ensure at least half of the likes are foreign

ORDER BY

foreign\_like\_ratio DESC; -- Sort by the foreign-like ratio in descending order

### **Explanation of Changes:**

1. **Added first\_foreign\_likers CTE**: This identifies the first foreign liker for each message by sorting likes based on their creation date.
2. **Joined person table**: Fetches the full name of the first foreign liker.
3. **Included like\_creation\_date**: Provides the timestamp of when the first foreign like occurred.

This query fulfills the new requirement and builds on the existing logic seamlessly.