

# Form items full width

Hello I have written the following sql query for the given schema could you please add some functionality:

Query:

```
select like_count.m_messageid, total_likes, foreign_likes
from
  (select count(*) total_likes, m_messageid from (select *
from message where m_length > 100) messages
  inner join likes on
messages.m_messageid=likes.l_messageid group by
m_messageid) as like_count
  inner join
  (select count(*) foreign_likes, m_messageid from
(select * from message where m_length > 100) messages
  inner join likes on
messages.m_messageid=likes.l_messageid where
  l_personid not in (select k_person2id from knows
where k_person1id = m_creatorid) and
  l_personid not in (select k2.k_person2id from knows k1,
knows k2 where k1.k_person1id = m_creatorid and
k1.k_person2id=k2.k_person1id)
  group by m_messageid)
as foreign_like_count
on
like_count.m_messageid=foreign_like_count.m_messageid
where like_count.total_likes>=20 and foreign_likes >=
total_likes/2 order by ((100*foreign_likes)/(total_likes)) desc;
```

Schema:

```
create table post (
  m_messageid bigint not null,
  m_ps_imagefile varchar,
  m_creationdate timestamp with time zone not null,
  m_locationip varchar not null,
  m_browserused varchar not null,
  m_ps_language varchar,
  m_content text,
  m_length int not null,
  m_creatorid bigint not null,
```

```

        m_ps_forumid bigint,
        m_locationid bigint not null
    );

create table comment (
    m_messageid bigint not null,
    m_creationdate timestamp with time zone not null,
    m_locationip varchar not null,
    m_browserused varchar not null,
    m_content text,
    m_length int not null,
    m_creatorid bigint not null,
    m_locationid bigint not null,
    m_replyof_post bigint,
    m_replyof_comment bigint
);

create table forum (
    f_forumid bigint not null,
    f_title varchar not null,
    f_creationdate timestamp with time zone not null,
    f_moderatorid bigint not null
);

create table forum_person (
    fp_forumid bigint not null,
    fp_personid bigint not null,
    fp_joindate timestamp with time zone not null
);

create table forum_tag (
    ft_forumid bigint not null,
    ft_tagid bigint not null
);

create table organisation (
    o_organisationid bigint not null,
    o_type varchar not null,
    o_name varchar not null,
    o_url varchar not null,
    o_placeid bigint not null
);

```

```
create table person (  
    p_personid bigint not null,  
    p_firstname varchar not null,  
    p_lastname varchar not null,  
    p_gender varchar not null,  
    p_birthday date not null,  
    p_creationdate timestamp with time zone not null,  
    p_locationip varchar not null,  
    p_browserused varchar not null,  
    p_placeid bigint not null  
);
```

```
create table person_email (  
    pe_personid bigint not null,  
    pe_email varchar not null  
);
```

```
create table person_tag (  
    pt_personid bigint not null,  
    pt_tagid bigint not null  
);
```

```
create table knows (  
    k_person1id bigint not null,  
    k_person2id bigint not null,  
    k_creationdate timestamp with time zone not null  
);
```

```
create table likes (  
    l_personid bigint not null,  
    l_messageid bigint not null,  
    l_creationdate timestamp with time zone not null  
);
```

```
create table person_language (  
    plang_personid bigint not null,  
    plang_language varchar not null  
);
```

```
create table person_university (  

```

```

    pu_personid bigint not null,
    pu_organisationid bigint not null,
    pu_classyear int not null
);

create table person_company (
    pc_personid bigint not null,
    pc_organisationid bigint not null,
    pc_workfrom int not null
);

create table place (
    pl_placeid bigint not null,
    pl_name varchar not null,
    pl_url varchar not null,
    pl_type varchar not null,
    pl_containerplaceid bigint -- null for continents
);

create table country (
    ctry_city bigint not null,
    ctry_name varchar not null
);

create table message (
    m_messageid bigint not null,
    m_ps_imagefile varchar,
    m_creationdate timestamp with time zone not null,
    m_locationip varchar not null,
    m_browserused varchar not null,
    m_ps_language varchar,
    m_content text,
    m_length int not null,
    m_creatorid bigint not null,
    m_locationid bigint not null,
    m_ps_forumid bigint,
    m_c_replyof bigint
);

create table message_tag (
    mt_messageid bigint not null,
    mt_tagid bigint not null

```

```
);

create table tagclass (
  tc_tagclassid bigint not null,
  tc_name varchar not null,
  tc_url varchar not null,
  tc_subclassoftagclassid bigint -- null for the root tagclass
  (Thing)
);

create table tag (
  t_tagid bigint not null,
  t_name varchar not null,
  t_url varchar not null,
  t_tagclassid bigint not null
);
```

Feature request:

“For each message, also return the full name of the first foreign person who liked the message and the time when they created the like.”

To add the requested functionality, we need to modify the query to include the first foreign person who liked each message along with their full name and the time they liked it.

We'll add an additional subquery that:

1. Identifies the first foreign person who liked each message (using the `MIN` function on the like creation timestamp).
2. Joins with the `person` table to retrieve the foreign person's full name and like creation timestamp.

Here's the updated query:

sql

```
SELECT
  like_count.m_messageid,
  total_likes,
  foreign_likes,
  first_foreign_person.p_firstname || ' ' || first_foreign_person.p_lastname AS
  first_foreign_person_name,
  first_foreign_like.l_creationdate AS first_foreign_like_time
```

```

FROM
  (SELECT
    COUNT(*) AS total_likes,
    m_messageid
  FROM
    (SELECT * FROM message WHERE m_length > 100) messages
  INNER JOIN
    likes ON messages.m_messageid = likes.l_messageid
  GROUP BY
    m_messageid) AS like_count
INNER JOIN
  (SELECT
    COUNT(*) AS foreign_likes,
    m_messageid,
    MIN(l_creationdate) AS first_foreign_like_time,
    l_personid AS first_foreign_person_id
  FROM
    (SELECT * FROM message WHERE m_length > 100) messages
  INNER JOIN
    likes ON messages.m_messageid = likes.l_messageid
  WHERE
    l_personid NOT IN (SELECT k_person2id FROM knows WHERE k_person1id =
m_creatorid)
    AND l_personid NOT IN (SELECT k2.k_person2id
                           FROM knows k1, knows k2
                           WHERE k1.k_person1id = m_creatorid
                              AND k1.k_person2id = k2.k_person1id)
  GROUP BY
    m_messageid) AS foreign_like_count
ON
  like_count.m_messageid = foreign_like_count.m_messageid
INNER JOIN
  person AS first_foreign_person
ON
  foreign_like_count.first_foreign_person_id = first_foreign_person.p_personid
INNER JOIN
  likes AS first_foreign_like
ON
  foreign_like_count.m_messageid = first_foreign_like.l_messageid
  AND foreign_like_count.first_foreign_person_id = first_foreign_like.l_personid
  AND foreign_like_count.first_foreign_like_time =
first_foreign_like.l_creationdate

```

#### WHERE

like\_count.total\_likes >= 20

AND foreign\_likes >= total\_likes / 2

#### ORDER BY

((100 \* foreign\_likes) / (total\_likes)) DESC;

## Explanation of Changes

### 1. New Columns in `foreign_like_count` Subquery:

- `first_foreign_like_time`: Uses `MIN(l_creationdate)` to find the earliest like timestamp from a foreign person.
- `first_foreign_person_id`: Retrieves the `l_personid` of the first foreign person who liked the message.

### 2. Join with `person` Table:

- Joins `first_foreign_person_id` with `person.p_personid` to get the `p_firstname` and `p_lastname` fields, forming the foreign person's full name.

### 3. Join with `likes` Table for Exact `first_foreign_like_time`:

- Ensures that we retrieve the exact `l_creationdate` for the earliest foreign like.

## Output

The query will return:

- `m_messageid`: The message ID.
- `total_likes`: Total likes for the message.
- `foreign_likes`: Likes from foreign persons.
- `first_foreign_person_name`: Full name of the first foreign person who liked the message.
- `first_foreign_like_time`: The timestamp of the first foreign like.

This query now provides the additional requested data alongside the original metrics.

I got an error: psql:smilla.sql:76: ERROR: column "likes.l\_personid" must appear in the GROUP BY clause or be used in an aggregate function  
LINE 22: l\_personid AS first\_foreign\_person\_id

please fix it

The error occurs because `l_personid` is neither grouped by nor aggregated in the subquery `foreign_like_count`. To fix this, we need to ensure that `first_foreign_person_id` is consistently derived alongside `first_foreign_like_time` by using `DISTINCT ON`, which allows us to select a single row per unique message, ordered by the creation date to find the "first" foreign like. This approach also eliminates the need for additional joins to find the exact first foreign like.

Here's the corrected query:

sql

```
SELECT
    like_count.m_messageid,
    total_likes,
    foreign_likes,
    first_foreign_person.p_firstname || ' ' || first_foreign_person.p_lastname AS
first_foreign_person_name,
    first_foreign_like_time AS first_foreign_like_time
FROM
    (SELECT
        COUNT(*) AS total_likes,
        m_messageid
    FROM
        (SELECT * FROM message WHERE m_length > 100) messages
    INNER JOIN
        likes ON messages.m_messageid = likes.l_messageid
    GROUP BY
        m_messageid) AS like_count
INNER JOIN
    (SELECT DISTINCT ON (m_messageid)
        m_messageid,
        COUNT(*) OVER (PARTITION BY m_messageid) AS foreign_likes,
        l_personid AS first_foreign_person_id,
        l_creationdate AS first_foreign_like_time
    FROM
        (SELECT * FROM message WHERE m_length > 100) messages
    INNER JOIN
        likes ON messages.m_messageid = likes.l_messageid
    WHERE
        l_personid NOT IN (SELECT k_person2id FROM knows WHERE k_person1id =
```



```

m_creatorid)
    AND l_personid NOT IN (SELECT k2.k_person2id
                           FROM knows k1, knows k2
                           WHERE k1.k_person1id = m_creatorid
                                AND k1.k_person2id = k2.k_person1id)

ORDER BY
    m_messageid, l_creationdate) AS foreign_like_count
ON
    like_count.m_messageid = foreign_like_count.m_messageid
INNER JOIN
    person AS first_foreign_person
ON
    foreign_like_count.first_foreign_person_id = first_foreign_person.p_personid
WHERE
    like_count.total_likes >= 20
    AND foreign_likes >= total_likes / 2
ORDER BY
    ((100 * foreign_likes) / (total_likes)) DESC;

```

## Explanation of Changes

### 1. Using `DISTINCT ON` in `foreign_like_count` Subquery:

- `DISTINCT ON (m_messageid)` allows us to select the first row for each `m_messageid`, ordered by `l_creationdate` to get the earliest foreign like.
- `COUNT(*) OVER (PARTITION BY m_messageid) AS foreign_likes` calculates the total foreign likes for each message without needing `GROUP BY`.

### 2. Removed Unnecessary Joins:

- With `DISTINCT ON`, we no longer need to join the `likes` table again to get `first_foreign_like_time`.

This updated query should now return the correct results without the grouping error.