SQL Query for Foreign Likes

I have a database with the following 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
);"
```

I needed to implement this query task:
“Find all messages longer than 100 characters that have been liked by at least 20 people. Only consider messages where at least half of the likes are from people that are foreign to the message’s creator, with a foreigner being a person that neither the creator nor any of their friends know. For each message, return the message’s id, the total number of likes and the number of foreign likes. Sort the returned messages by the ratio of foreign likes from highest to lowest.”
For this I generated this query:

```
"select m_messageid, total_likes, foreign_likes
from message
left join lateral (
  select l_messageid, count(*) as foreign_likes
  from likes
  where
    l_personid not in (
      select level1friend.k_person2id
      from knows as level1friend
      where level1friend.k_person1id=m_creatorid) and
    l_personid not in (
      select level2friend.k_person2id as friendfriendid
      from knows as level2friend
      where level2friend.k_person1id in (
        select level1friend.k_person2id
        from knows as level1friend
        where level1friend.k_person1id=m_creatorid))
  group by l_messageid
) fl on message.m_messageid=fl.l_messageid
left join (
  select l_messageid, count(*) as total_likes
  from likes
  group by l_messageid
) tl on message.m_messageid=tl.l_messageid
where m_length>100 and total_likes>19 and foreign_likes>=cast(total_likes AS float)/2
order by (foreign_likes/cast(total_likes as float)) desc;"
```

now I need to expand it to also cover this following task;
"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"
Expand the given query

To expand the query to include the **full name of the first foreign person** who liked the message and the **time when they created the like**, you need to:

1. Add a subquery or lateral join to retrieve the first foreign like for each message.

2. Include the `person` table to get the full name of the person who liked the message.
3. Modify the SELECT clause to include these additional details.

Here's the updated query:

```sql
select
    message.m_messageid,
    total_likes,
    foreign_likes,
    first_foreign_like.liker_fullname,
    first_foreign_like.like_creationdate
from message
left join lateral (
  select l_messageid, count(*) as foreign_likes
  from likes
  where
    l_personid not in (
      select level1friend.k_person2id
      from knows as level1friend
      where level1friend.k_person1id = message.m_creatorid) and
    l_personid not in (
      select level2friend.k_person2id as friendfriendid
      from knows as level2friend
      where level2friend.k_person1id in (
        select level1friend.k_person2id
        from knows as level1friend
        where level1friend.k_person1id = message.m_creatorid))
  group by l_messageid
) fl on message.m_messageid = fl.l_messageid
left join (
  select l_messageid, count(*) as total_likes
  from likes
  group by l_messageid
) tl on message.m_messageid = tl.l_messageid
left join lateral (
  select
    person.p_firstname || ' ' || person.p_lastname as liker_fullname,
    likes.l_creationdate as like_creationdate
  from likes
  join person on likes.l_personid = person.p_personid
  where
    likes.l_messageid = message.m_messageid and
    person.p_personid not in (
      select level1friend.k_person2id
      from knows as level1friend
      where level1friend.k_person1id = message.m_creatorid) and
    person.p_personid not in (
      select level2friend.k_person2id as friendfriendid
      from knows as level2friend
      where level2friend.k_person1id in (
        select level1friend.k_person2id
        from knows as level1friend
        where level1friend.k_person1id = message.m_creatorid))
  order by likes.l_creationdate
  limit 1
) first_foreign_like on true
where
    m_length > 100
    and total_likes > 19
    and foreign_likes >= cast(total_likes as float) / 2
order by
    (foreign_likes / cast(total_likes as float)) desc;
```

Explanation of Additions

1. **`first_foreign_like` Lateral Join:**

- This subquery retrieves the full name and like creation date of the first foreign person who liked the message (`` `order by likes.l_creationdate limit 1` ``).
    - It uses the same logic to filter foreign persons as the `` `foreign_likes` `` computation.
2. `` `person` `` **Table Join**:
    - Used to get the `` `p_firstname` `` and `` `p_lastname` `` for the person who liked the message.
3. **SELECT Clause**:
    - Includes `` `first_foreign_like.liker_fullname` `` and `` `first_foreign_like.like_creationdate` ``.

Result

The query now returns:

- The message ID.
- The total number of likes.
- The number of foreign likes.
- The full name of the first foreign liker.
- The time when they created the like.