1. Using the data schema in table 1, write a query that would bucket the users based on the following criteria:

A user cannot be in more than 1 bucket
Date: Jan 1, 2016 to Jan 7, 2016
The buckets are prioritized below:
1. Priority 1: only users who made contact on 2016-01-01
2. Priority 2: exchanged more than 50 messages
3. Priority 3: interacted with more than 30 other users

Assumptions:

- **A user cannot be in more than 1 bucket.**

//This user is not delimiter clearly, but I treat this both for host and guest.

- **Date: Jan 1, 2016 to Jan 7, 2016**

// (1) date not co Jan 7, 2016.
But I treat it as >=

- **The buckets are prioritized below:**

1. Priority 1: only users who made contact on 2016-

//there are 2 understandings,
(1) This could be the case that if user made contact on 2016-01-01 then the user is in the scope.
(2) The user only made contact on 2016-01-01 no any chat out of 2016-01-01, then the user in the scope.
I pick (1) as my assumption.

2. Priority 2: exchanged more than 50 messages.

The table1's num_messages should count all the interaction messages between host and guest in the same day, but there are still 2 understanding of the 50 messages.
(1) The user exchanged more than 50 messages in total from Jan 1, 2016 to Jan 7, 2016.
(2) The user exchanged more than 50 messages in any of the chat between Jan 1, 2016 to Jan 7, 2016.
I pick (1) as my assumption.

3. Priority 3: interacted with more than 30 other users

//this part is a trick part, and there is no definition of the tables and business rules.
But for normal business, the system should not allow id_host and id_guest are using the same id and the same time all the host and guest id cannot be null.

But even so, interacted with more than 30 other users can be understand in 2 ways:

(1) The user (could be host or guest) has chat with 30 host or guest (rows in the table) from Jan 1, 2016 to Jan 7, 2016.
   This means the guest could chat with the same host for 3 days and could chat few messages for each day. Then the interacted with users only count as 3.

(2) The user (could be host or guest) has chat with 30 DIFFERENT host or guest from Jan 1, 2016 to Jan 7, 2016.
   This means the guest could chat with the same host for 3 days and could chat few messages for each day. But the interacted with users only count as 1.

Both could be correct, so I write 2 script, but I prefer (2) as better business understanding.

Test data:

TestData.sql

Whole Script:

Priority3 using case (1)

SET group_concat_max_len=100000000;

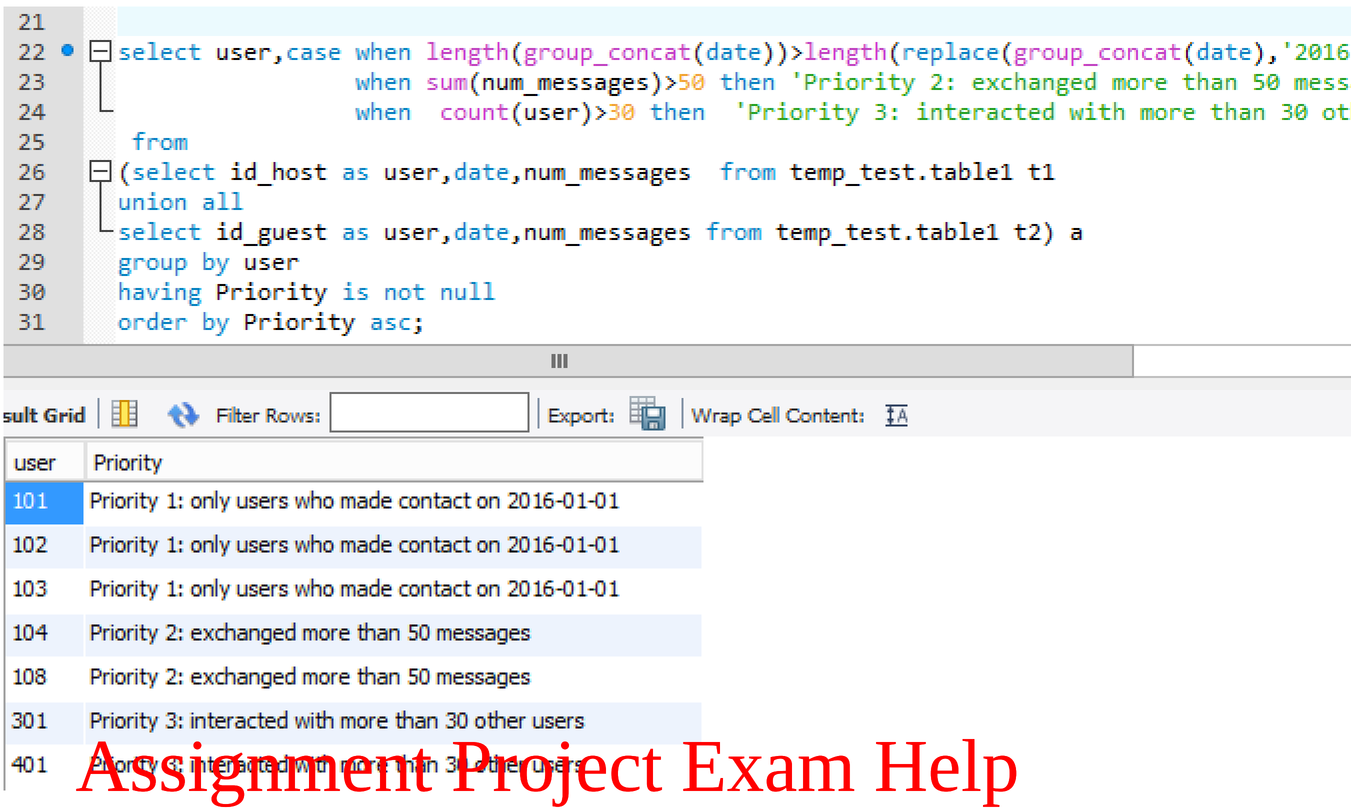
select user,case when length(group_c [illegible] 016-01-01','')) then 'Priority 1: only [illegible]
when sum(num_messages)>5 [illegible] ed more than 50 messages [illegible]
when count(user)>30 then 'Prio [illegible] ore than 30 other users' end as Priority
 from
(select id_host as user,date,num_messages from temp_test.table1 t1
union all
select id_guest as user,date,num_messages from temp_test.table1 t2) a
group by user
having Priority is not null
order by Priority asc;

#Logic:

(1)length(group_concat(date))>replace(group_concat(date) if this is true, then it means date filed has date 2016-01-01

(2)sum(num_messages)>50 if this is ture that means the exchanged messages more than 50.

(3)count(user)>30 if this is true that means interacted with more than 30 other users



```
select user,case when length(group_concat(date))>length(replace(group_concat(date),'2016
                 when sum(num_messages)>50 then 'Priority 2: exchanged more than 50 mess
                 when  count(user)>30 then  'Priority 3: interacted with more than 30 ot
 from
(select id_host as user,date,num_messages  from temp_test.table1 t1
union all
select id_guest as user,date,num_messages from temp_test.table1 t2) a
group by user
having Priority is not null
order by Priority asc;
```

user	Priority
101	Priority 1: only users who made contact on 2016-01-01
102	Priority 1: only users who made contact on 2016-01-01
103	Priority 1: only users who made contact on 2016-01-01
104	Priority 2: exchanged more than 50 messages
108	Priority 2: exchanged more than 50 messages
301	Priority 3: interacted with more than 30 other users
401	Priority 3: interacted with more than 30 other users

Priority3 using

SET group_concat [illegible]

select user,

case when find_in_set('2016-01-01',group_ [illegible] rity 1: only users who made contact on 2016-01-01

when sum(num_messages)>50 then 'Priority 2: exchanged more than 50 messages'

when count(user)>30 then 'Priority 3: interacted with more than 30 other users' end as Priority

from

((select id_host as user,group_concat(date) as date,sum(num_messages) as num_messages from temp_test.table1 group by id_host,id_guest)

union all

(select id_guest as user,group_concat(date) as date,sum(num_messages) as num_messages from temp_test.table1 group by id_host,id_guest))a

group by user

having Priority is not null

order by Priority asc;

#Logic:

(1) find_in_set('2016-01-01',group_concat(date)) if this is true, then it means date filed has date 2016-01-01

(2)sum(num_messages)>50 if this is ture that means the exchanged messages more than 50.

(3)count(user)>30 if this is true that means interacted with more than 30 other users

2. Using table 2, how would you go about t

assumption that there can only be one valid tax_area_id

per listing?

Assumption:

(1) This is MySQL DB

(2) We can check the table definitions

(3) Listing means id_listing in the table2

Solutions:

method1:

Step1: show create table2;

Step2: if id_listing and tax_area_id are built as joint primary key then no need check, Databse already has restriction on this.

method2:
if there is no primary key for (id_listing,tax_area_id)
select * from table2 group by id_listing having count(distinct tax_area_id)>1;
if retrun 0 rows then there can only be one valid tax_area_id per listing.

3. Using table 2, assuming that there are duplicate entries per listing, write a query that only returns the latest valid information entered per listing.

Assumption:
(1) deleted_at fil [illegible]
(2) Created_at al [illegible]

#Script
select * from (select * from table2 where deleted_at is [illegible] y create_at desc) a group by id_listing;

#Logic:
1. filter the deleted_at filed make sure the record is not deleted.
2. order by the rest of the records by field create_at sort by desc, then group by the id_listing to make sure we only fetch the first row for each id_listing, that means the latest vlaid informaiton.