--categories
-- 1. Spam firend request
-- 2. dwell time/usage
-- 3. post
-- 4. Video(call/portal)
-- 5. SMS verification
-- 6. workplace
-- 7. shop
-- 8. Friendship
-- 9. Message
--10. Ads
--11. Spam
--12. Composer
--13. Search
--14. fundrasier
--15. Video(watch)
--16. Feature
--17. Other

-- ----------1. Spam Friend Request----------
friending:
sender_id | receiver_id | send_time | accept_time
users:
user_id| spam_type ("fraud", etc)

-- 1.上周每一天的 same day accept rate

```
select
        date(send_time),
        sum(date(send_time) = date(accept_time))/count(*) as same_day_rate
from friending
where datediff(curdate(),date(send_time)) <= 7
group by 1
```

-- 2.上周所有的 request 里，不是由 spam 用户发出的比例

```
select
        sum(case when u.spam_type = 'fraud' then 0 else 1 end)/
        count(*) as not_spam_rate
from friending f
```

```
left join users u
on f.sender_id = u.user_id
where datediff(date(send_time), Curdate()) <= 7


-- ----------------------2. dwell time/usage-------------------------------
Table1: user_sessions
date | user_id | start_time | end_time | end_reason
-------------------------------------------------------------------
'2020-09-01' | 1 | 1964783746 | 1964783924 | 'close-app'
'2020-09-01' | 1 | 1964783528 | 1964783809 | 'crash'
'2020-09-02' | 2 | 1964783123 | 1964783345 | 'close-app'
'2020-09-02' | 3 | 1964783252 | 1964783658 | 'crash'

Table2: dim_all_users
user_id | country | is-active
-----------------------------------
1 | 'US' | TRUE
2 | 'US' | FALSE
3| 'CANADA' | TRUE

-- 1. What is the average time (in seconds) each user stays on Facebook on specific date?
-- | user_id | avg(time)

select
    user_id,
    avg(end_time - start_time) as avg_time
from user_sessions
where date = 'xxxx-xx-xx'
group by 1

-- 2. What is the percentage of active users who crashed when using Facebook from the
US area on a specific date?
-- | ratio |
-- 1. filter US & date & active 2,user_crash/all

select
        count(distinct s.user_id)/count(distinct u.user_id) as ratio
from dim_all_users u
join user_sessions s
on u.user_id = s.user_id
```

```
and s.end_reason = 'crash'
where u.country = 'US'
and u.is_active is True
and date = 'xxxx-xx-xx'

-----------------------------------------------------------------------
user_event: User_id | event_time | event_name | session id

                                                    |    open menu
                                                    |    close menu
-- 1. Calculate the average dwell time in seconds across all sessions (i.e. return one number)?
-- Dwell time is the length of time between opening and closing the menu.


-- assume: for each session, the first event_name is open and then followed by close
select
      avg(lead(event_time,1) over(partition by user_id, session_id order by event_time asc) - event_time) as avg_dwell
from user_event

-- assume: for each session, there are other types of event_name as well
select
      avg(c.event_time - o.event_time) as avg_dwell -- avg will ignore null
from user_event o
left join user_event c
on o.user_id = c.user_id
and o.session_id = c.session_id
and c.event_name = 'close menu'
where o.event_name = 'open menu'

-- 2. Get the percentage of all sessions that have both nav_menu_open and nav_menu_close?
-- assume: all have open, but some do not have close
select
      count(distinct c.session_id)/count(o.session_id) as percent_both
from user_event o
left join user_event c
on o.user_id = c.user_id
and o.session_id = c.session_id
and c.event_name = 'close menu'
```

```
where o.event_name = 'open menu'

-- assume: either open and close can be missing
select
    sum(case when nav_menu_open is not null and nav_menu_close is not null then 1 else
0 end)/
    count(distinct session_id) as ratio
from
    (select distinct
        a.session_id,
        b.event_name as nav_menu_open,
        c.event_name as nav_menu_close
    from user_event a
    left join user_event b
    on a.user_id = b.user_id
    and a.session_id = b.session_id
    and b.event_name = 'nav_menu_open'
    left join user_event c
    on a.user_id = c.user_id
    and a.session_id = c.session_id
    and c.event_name = 'nav_menu_close')s


-- 3. Lets say we want to account for missing events by setting the dwell time to 60 seconds
whenever a nav_menu_close event is missing.
-- Can you write a query to re-calculate the new average dwell time when we default to
60 seconds of dwell time whenever nav_menu_close is missing?

select
        avg(ifnull(c.event_time - o.event_time),60) as avg_dwell -- avg will ignore null
from user_event o
left join user_event c
on o.user_id = c.user_id
and o.session_id = c.session_id
and c.event_name = 'nav_menu_close'
where o.event_name = 'nav_menu_open'
-- v2 use subquery instead of join on and
select
        avg(ifnull(b.event_time - a.event_time,60)) as avg_dwell
from user_event a
left join (select *
```

```
        from user_event
        where event_name = 'nav_menu_close'
    ) b
on a.session_id = b.session_id
and a.user_id = b.user_id
where a.event_name = 'nav_menu_open'

-- 4. Calculate the average gap between each session
-- assume gap is the difference between open time - previous close time
select
    avg(gap) as avg_gap
from
    (select
        event_name,
        timestampdiff(second,
                      lead(event_time,1) over(partition by user_id order by
event_time asc),
                      event_time) as gap
    from user_event)s
where event_name = 'close menu'
```

每次登陆一个 app 算一个 session , user ID 有重复, session ID unique, 有每个 session

的起始时间和结束时间

```
user_sessions
    date       |user_id | session_id | start_time | end_time    | app
-------------------------------------------------------
 '2020-09-01' |    1     |      1       |1964783746 | 1964783924 | 'FB'
 '2020-09-01' |    1     |      2       |1964783528 | 1964783809 | 'Ins'
 '2020-09-01' |    1     |      5       |1964784009 | 1964784109 | 'FB'
 '2020-09-02' |    2     |      3       |1964783123 | 1964783345 | 'Messenger'
 '2020-09-02' |    3     |      4       |1964783252 | 1964783658 | 'FB'
```

q1: 每个 app average daily performance 如何, 哪个比较好, 先问你会定个什么 metric

来算, 然后写这个 average daily 的

```
-- DAU and TS/DAU per day
select
```

```
    date,
    app,
    count(distinct user_id) as dau,
    sum(timediff(end_time, start_time))/count(distinct user_id) as ts_dau
from user_sessions
group by 1,2

-- # average daily users
select
    app,
    avg(ct_user) as avg_daily_user
from
    (select
        date,
        app,
        count(distinct user_id) as ct_user
    from user_sessions
    group by 1,2)s

-- # average daily ts per user
select
    app,
    avg(ts_per_user) as avg_daily_ts
from
    (select
        date,
        app,
        sum(timediff(end_time, start_time))/
        count(distinct user_id) as ts_per_user
    from user_sessions
    group by 1,2)s
```

q2：算平均每个 app 用户每次使用 app 之间的时间差

```
-- start - previos end
select
    app,
    avg(timediff(
        start_time,
        lag(end_time,1) over(partition by user_id, app order by start_time asc))) as
avg_gap
```

```
from user_sessions
group by 1
```

q3: 算 每天每个 app bounce rate，就是从 app1 -> app 2-> 再回到 app1

补充一些 sql 我的思路，因为 bounce rate 这里定义比较简单就是 A -> B -> A，

就是必须第二次重新回到 a 就是一次 bounce

```
-- assume: # bounce / # open

select
	date,
	sum(case when (app = thrid and app <> next) then 1 else 0 end)*1.00/count(*) as
bounce_rate
from
(select
	date,
	app,
	lead(app,1) over(partition by date, user_id order by start_time asc) as next,
	lead(app,2) over(partition by date, user_id order by start_time asc) as thrid
from user_sessions)s
group by 1
```

--Session table: Date | sessionid | userid | action (enter/click/send/exit)

--Time table （注意，这里的 sessionid 都是 unique 的: Date | Sessionid | time_spent (s)

```
-- Q1: what is the average number of sessions per user for the past 30 days
select
	count(sessionid)/
	count(distinct userid) as session_per_user
from session
where date >= date_add(Curdate(), inverval - 30 day)
-- Q2: # of users who at least spent more than 10s on each session
select
	count(distinct userid) as num_user
from session
where userid not in
	(select
```

```
        s.user_id
    from session s
    join time t
    on s.sessionid = t.sessionid
    where t.time_spent < 10)
-- Q3: Time distribution of each user (Exponential Distribution)
-- Q; per user or per user per day?
select
    ts,
    count(userid) as frequency
from
    (select
        s.userid,
        sum(time_spent) as ts
    from session s
    join time t
    on s.sessionid = t.sessionid
    group by 1)s
group by 1
-- Q4: daily active user for the past 30 days
-- V1. Q: for the past 30 days, number of user who use facebook everyday?
select
    count(distinct userid) as total_dau
from session
where date between datediff(date, interval -30 days) and datediff(curdate(), interval -1
days)
group by 1
hving count(distinct date) = 30
```

SQL

给了一个 app session table

Columns:
userid (varchar)
app ['fb', 'messenger', 'IG'] (string)
interface ['andriod', 'iphone', 'web']
sessionid (varchar)
session_start_time (int)
session_end_time (int)

1. 哪个 app 在昨天用户访问量最多

```
select
	app,
	count(distinct userid) as ct
from app_session
where date(session_start_time) = date_add(curdate(), interval -1 day)
group by 1
order by 2 desc
limit 1
-- in case tie
select
	app
from
	(select
		app,
		rank() over(partition by app order by count(distinct userid) desc) as rk
	from app_session
	group by 1)s
where rk = 1
```

2. 对于有多个 session 的用户，平均访问时间间隔是多少？

```
select
	userid,
	avg(lead(session_start_time,1) over(partition by userid order by session_start_time
asc) - session_end_time) as gap
from app_session
group by 1
```

---

usage:

```
date_time | volume | platform | country
--------------------+--------+----------+---------
2017-01-01 00:00:00 | 231 | iOS | Finland
2017-01-01 00:00:00 | 446 | iOS | Belgium
2017-01-01 00:00:00 | 353 | iOS | Italy
2017-01-01 00:00:00 | 342 | Android | Finland
2017-01-01 00:00:00 | 468 | Android | Belgium
2017-01-01 00:00:00 | 296 | Android | Italy
2017-01-02 00:00:00 | 461 | iOS | Finland
2017-01-02 00:00:00 | 382 | iOS | Belgium
2017-01-02 00:00:00 | 357 | iOS | Italy
2017-01-02 00:00:00 | 224 | Android | Finland
```

2017-01-02 00:00:00 | 216 | Android | Belgium
2017-01-02 00:00:00 | 331 | Android | Italy

```
-- 1. Find the absolute differences in day-by-day volume between iOS and Android, ignoring countries
with cte as (
select
	date(date_time),
	platform,
	sum(volumn) as sum_volumn
from usage
group by 1,2)

select
	c1.dtae,
	c1.platfrom,
	c2.platfrom,
	c1.volumn - c2.volumn as diff
from cte1 c1
left join cte2 c2
on c1.date = c2.date
and c2.platform = 'Android'
where c1.platfrom = 'iOS'

-- 2. Find the absolute differences in day-by-day volume for every pair of countries (without repeating pairs), ignoring platforms
with cte as (
select
	date(date_time),
	country,
	sum(volumn) as sum_volumn
from usage
group by 1,2)

select
	c1.dtae,
	c1.country,
	c2.country
	c1.volumn - c2.volumn as diff
from cte1 c1
left join cte2 c2
on c1.date = c2.date
```

```
and c1.country < c2.country

-- ----------------------------------------------3. Posts----------------------------------
comments: user_id | ds | comments
ds --# date
comments -- # of comments posted by the user
user:user_id | country -- country where the user register at

-- 1. return the total number of posts per user in May 2020
-- count(post)/count(user_id)
-- filter: may & 2020
-- 1. join 2. count post in 2020/5 3. divide by # user

select
    count(comments)/
    count(distinct user_id) as post_per_user
    from user u
left join comment c
on u.user_id = c.user_id
and year(c.ds) = 2020 and month(c.ds) = 5 -- assume the denominator is [all user] regarless
of the time[confirm with interviewer]

-- 2. return the percentage of users that posted comments in May 2020

select
    count(distinct c.user_id)/
    count(distinct u.user_id) as ratio
    from user u
left join comment c
on u.user_id = c.user_id
and year(c.ds) = 2020 and month(c.ds) = 5

-----------------------------------------------------------------------------------------------
Table: user_actions
date | actor_id | post_id | relationship | interaction
Table: user_posts
date | poster_id | post_id
-- 1. How many likes were made on Friend posts yesterday?
select count(*) as n_likes
from user_actions
where relationship = 'friend'
```

```
and interaction = 'like'
and `date` = date_add(Curdate(), interval -1 day)


-- 2. If I were user 123, how would you calculate the average number of likes on all of my
posts?
-- 1. # post 2. # likes -- left join with on
select
    count(a.interaction)/count(distinct p.post_id) as likes_per_post
from user_posts p
left join user_actions a
on p.post_id = a.post_id
and interaction = 'like'
where poster_id = '123'

---------------------------------------------------------------------
Content_action: | date | user_id | content_id | content_type | target_id |
User_id (content_creator_id)
Content_id (this is the primary key),
Content_type (with 4 types: status_update, photo, video, comment),
Target_id (the original content_id associated with the comment, if the content type is not
comment, this will be null)


-- 1.find the distribution of stories（photo+video） based on comment count?

-- # of comments | # of stories (photo+video)
-- 1. # comments per post 2. group by # comments count # posts
select
    ct,
    count(content_id) as frequency
from
    (select
        o.content_id,
        count(distinct c.content_id) as ct
    from content_action o
    left join content_action c
    on o.content_id = c.target_id
    and c.content_type = 'comment'
    where o.content_type in ('photo' and 'video')
    group by 1)s
group by 1
```

```
-- 2.Now what if content_type becomes {comment, post, video, photo, article}，what is
the comment distribution for each content type ?
-- | content_type | number of comments | frequency
-- I guess comment will not have comment, since target_id is the original content_id
-- content_id | type| # comments ->    group by type and # comments, count(content_id)

select
      content_type,
      ct,
      count(content_id) as frequency
from
      (select
            a.content_id,
            a.content_type,
            count(b.countent_id) as ct
      from Content_action a
      left join Content_action b
      on a.content_id = b.target_id
      and b.content_type = 'comments'
      where and a.target_id is not null
      group by 1)s

------------------------------4. Video(call/portal)----------------------
video_calls:
caller| recpient| ds| call_id| duration
fb_dau:
user_id| DAU_flag| ds| country
--Q1: On 2020-01-01 how many people initiated multiple calls?
select
      count(caller) as ct
from
      (select
            caller
      from video_calls
      where ds = '2020-01-01'
      group by 1
      having count(distinct call_id) >1)s;
```

```
--Q2: % of DAU used the video calls function by each country on 2020-01-01?
-- | country | % |
-- filter: 1. filter on 2020-01-01 2. DAU
-- group by country
-- 1. join on caller/recpient 2.DAU 3.count
select
	country,
	1 - sum(case when v.caller is null then 1 else 0 end)*1.00/count(distinct u.user_id) as
percent_dau
from fb_dau u
left join video_calls v
on (u.user_id = v.caller or u.user_id = v.recpient)
and u.ds = v.ds
where u.ds = '2020-01-01'
and DAU_flag = 1
group by 1

-- v2
select
	country,
	sum(tag)/count(*) as percent_dau
from
	(select distinct
		country,
		f.user_id,
		case when v.caller_id is not null or v.recpient is not null then 1 else 0 end as tag
	from fb_dau f
	left join video_calls v
	on f.ds = r.ds and (f.user_id = v.caller or f.user_id = r.recpient)
	where f.ds = '2020-01-01'
	and DAU_Flag    = 1)s
group by 1;
-------------------------------------------------------------------------
Table:
timestamp, callerid, receiverid, call length

-- 1. 找出在 20190101 这一天的新用户。新用户可以是打过电话的人，也可以是接
过电话的人

select
	user
```

```
from
    (select
        callerid as user,
        date(timestamp) as date
    from table
    union
    select
        receiverid as user,
        date(timestamp) as date
    from table)s
group by 1
having min(first_date) = '2019-01-01'

-- 2. 对这一天的新用户，算出之后每一天的 retention rate，直到今天，例如

day, retention rate
1, 0.9
2, 0.8

with c1 as (
    select
        user,
        min(date) as day
    from
        (select
            callerid as user,
            date(timestamp) as date
        from table
        union
        select
            receiverid as user,
            date(timestamp) as date
        from table)s
    group by 1
    having min(date) = '2019-01-01')
c2 as (
    select
        callerid as user,
        date(timestamp) as date
    from table
    union
```

```
    select
        receiverid as user,
        date(timestamp) as date
    from table)

select
    datediff(c2.day, c1.day) as day,
    count(distinct c2.user)/count(distinct c1.user) as retention_rate
from c1
left join c2
on c1.user = c2.user
and c1.day < c2.day
group by 1
```

----------5. SMS Verification----------

sms_message (fb to users)

| date | country | cell_numer | carrier | type

|2018-12-06 | US | xxxxxxxxxx | verizon | confirmation (ask user to confirm)

|2018-12-05 | UK | xxxxxxxxxx | t-mobile | notification

confirmation (users confirmed their phone number)

|date | cell_number |

(User can only confirm during the same day FB sent the confirmation message)

```
-- Q1. How many requests have we sent to each carrier and country?
select
    country,
    carrier,
    count(*) as num_request
from sms_message
where type = 'confirmation'
group by 1,2
-- Q2. Number of users who received notification every single day during the last 7 days
-- Q: cell_number as id?

select
    count(cell_number) as num_user
from
    (select
        cell_number
    from sms_message
    where type = 'notification'
```

```
        and date >= date_add(Curdate(), interval -7 day)
        and date <= date_add(Curdate(), interval -1 day)
        group by 1
        having count(distinct date) = 7)s

-- Q3. confirmation rate over the last 30 dayssele
select
        s.date,
        count(c.cell_number)/
        count(s.cell_number) as confirm_rate
from sms_message s
left join confirmation c
on s.cell_number = c.cell_number
and s.date = c.date
where s.type = 'confirmation'
and datediff(curdate(), date) <= 30
group by 1

-- Q4. On dec 06th, overall confirmation rate.
-- same thing
------------------------------6. Work Place------------------------------
Region | country | company_id | no_of_onboarded_employee
-- Q1. Calculate the average number of onboarded employees per country per region
select
        sum(no_of_onboarded_employee)/count(distinct concat(Region,country)) as ratio
from workspace

-------------------------------7. Facebook Shop-------------------------
date | order_id | sender | timestamp, sender = 1: seller, sender=0: buyer
-- Q1. how many orders have message from buyer on date xxxx.
-- |# order|
select
        count(distinct order_id) as ct
from order
where sender = 0
and date = 'xxxx'
-- how many percent of order have message from buyer on date xxxx?
select
        sum(sender = 0)/count(distinct order_id)
from
        (select distinct
```

```
            order_id,
            sender
        from order
        where `date` = 'xxx-xx-xx')s
-- Q2. how many orders have messages both from buyer and seller.
select
        count(order_id) as ct
from
        (select
            order_id
        from order
        group by 1
        having count(discint sender) = 2)s
-- if there are other type, use the following version
select
        count(distinct order_id) as ct
from order o
join (select
            order_id
          from order
          where sender = 0)s
on o.order_id = s.order_id
where o.sender = 1
-- Q3. how many orders with last message sent from seller.
select
        count(order_id) as ct
from
        (select
            order_id,
            sender,
            row_number() over(partition by order_id order by timestamp desc) as rn
        from order)s
where rn = 1
and sender = 1

-----------------------------------8. Friendship---------------------------
-- Friending: send_id || receive_id || send_time || accept_time || country
-- Age: user_id || age_group
-- 1) Same-day acceptance rate in the last 7 days
select
        sum(case when date(send_time) = date(accept_time) then 1 else 0 end)/
```

```
        count(*) as same_day_rate
from Friending
-- 2) Average requests sent per user for each age group
select
        age_group,
        avg(num_request) as avg_num_request
from
        (select
                a.user_id,
                a.age_group,
                count(*) as num_request
        from Age a
        left join Friending f
        on a.user_id = f.send_id
        group by 1,2)s
group by 1
-- other calculation

select
        age_group,
        count(distinct concat(send_id, receive_id))/
        count(distinct a.user_id)
from Age a
left join Friending f
on a.user_id = f.sned_id
group by age_group

-- https://www.1point3acres.com/bbs/thread-429467-1-1.html
-- friending: | ds | action | actor_uid | target_uid |
--ds = date
--action = {'send_request', 'accept_request', 'unfriend'}
--actor_uid = uid of person pressing the button to take the action
--target_uid = uid of the other person involved

-- /* Q1: How is the overall friending acceptance rate changing over time? */
-- v1: How is the overall friending acceptance rate changing over time?
-- acceptance rate per day
select
        r.ds,
        count(a.actor_uid)*1.00/count(r.action_uid) as accept_rate
from friending r
```

```
left join friending a
on r.target_uid = a.actor_uid
and a.action = 'accept_request'
where r.action = 'send_request'
group by 1

-- v2. Define how long you have to wait before a friend request is considered rejected (e.g. 1 week).
-- I guess the questions is : if a request is not been accepted within in one week, then we considier it as rejected. Calculate accpetance rate.
--1. self join: left(request), right(accptced), reverse id, one week
select
    round(sum(case when datediff(a.date, r.date) <=7 then 1 else 0 end)*100.00/
    count(*),2) as acceptance_rate
from friending r
left join friending a
on r.actor_uid = a.target_uid
and r.target_uid = a.actor_uid
and a.action = 'accept_request'
where r.action = 'send_request'

/* Q2: Who has the most number of friends?*/
with c1 as (
select distinct
    actor_uid as user1,
    target_uid as user2
from friending
where action = 'accept_request'

union all

select distinct
    target_uid as user1,
    actor_uid as user2
from friending
where action = 'accept_request'),
c2 as(
select distinct
    actor_uid as user1,
    target_uid as user2
from friending
```

```
where action = 'unfriend'

union all

select distinct
        target_uid as user1,
        actor_uid as user2
from friending
where action = 'unfriend'
)

select
        user1,
        count(distinct user2) as ct
from c1
where (user1, user2) not in (select * from c2)
group by 1
order by 2 desc
limit 1


-- use minus operator( does not support by mySQL)

with cte as (
        select
                actor_uid,
                target_uid
        from friending
        where action = 'accept_request'

        union all

        select
                actor_uid as target_uid,
                target_uid as actor_uid
        from friending
        where action = 'accept_request'

        MINUS

        select
```

```
        actor_uid,
        target_uid
    from friending
    where action = 'unfriend'

    MINUS

    select
        actor_uid as target_uid,
        target_uid as actor_uid
    from friending
    where action = 'unfriend')

select
    actor_uid
from
    (select
        actor_uid,
        rank() over(order by count(target_uid) desc) as rk
    from cte
    group by 1)s
where rk = 1;
-------------------------------------------------------------------------------
-- Q1: For each month in 2020, what is the friend request acceptance rate?
-- month (%Y - %m) | accptance rate (accpted/all)
-- 1. count, 2. divide 3.gorup by month
select
    date_format(date, '%Y-%m') as `date`,
    sum(result = 1)/ count(*) as acceptance_rate
from friend_requests
Group by 1

-- Q2: What percent of users in the United States accepted a friend request last week?
-- |ratio|
-- filter: USA, last week
-- 1. join 2. filter: USA, last week 3. count

select
    sum(result = 1) / count(distinct u.user_id) as user_acceptance
from
    (select distinct
```

```
        u.user_id,
        r.result
    from users u
    left join friend_requests r
    on u.user_id = r. receiver_id
    and datediff(CURDATE(),r.date) <= 7 -- use and to not remove users who didn't receive request last week
    where u.country= 'US')s
```

---

2 张表 user_network_requests 和 user_country.

success 代表发送好友请求是否通过，1 为通过，0 为没通过。

user_network_requests

userid	timestamp	data_center	success
10032	15009	A	1
10032	15097	C	0

user_country

userid	country
10032	US

1.求每个 data center，request fail 的比率

```
select
    data_center,
    1 - sum(success)*1.00/count(*) as fail_rate
from user_network_requests
group by 1
```

2.求每个国家，request fail 的比率

```
select
    c.country,
    1 - sum(success)*1.00/count(*) as fail_rate
from user_network_requests r
join user_country c
on r.userid = c.userid
group by 1
```

3.求每个国家，有多少个 user 发出的好友请求从来没有 fail 过

```
select
    country,
    count(userid) as ct
from user_country
where (userid, country) not in
(select distinct
    country,
    userid
from user_network_requests r
join user_country c
on r.userid = c.userid
where success = 0)
group by 1
--------------------------9. message--------------------------
-- | date | timestamp | senderid | receiverid | reaction flag | message_thread
-- Q1 unique conversation created
select
    count(distinct concat(user_1,user_2 )) as ct
from
    (select
        case when senderid < receiverid then senderid else receiverid end as user_1,
        case when senderid < receiverid then receiverid else senderid end as user_2
    from conversations)s

-- Q2 percentage of conversations has at least 1 reaction
-- 1. total conversation 2. conversations with reaction
with cte as (
select distinct
    case when senderid < receiverid then senderid else receiverid end as user_1,
    case when senderid < receiverid then receiverid else senderid end as user_2,
    reaction_flag
    from conversations)
select
    sum(flag = 'Y')/count(distinct concat(user1, user2)) as prct
from cte

-- Q3 average num of days between when the conversation started to 1st reaction
--- 1. start date 2. start of the 1st reaction
with
```

```
cte1 as (
select
    case when senderid < receiverid then senderid else receiverid end as user_1,
    case when senderid < receiverid then receiverid else senderid end as user_2,
    min(`date`) as first_date
from conversations
group by 1,2),
cte2 as (
select
    case when senderid < receiverid then senderid else receiverid end as user_1,
    case when senderid < receiverid then receiverid else senderid end as user_2,
    min(`date`) as first_date
where reaction_flag = 'Y'
from conversations
group by 1,2)

select
    avg(datediff(c2.first_date, c1.first_date)) as avg_date
from cte1 c1
join cte2 c2
where c1.user_1 = c2.user_1
and c1.user_2 = c2.user_2


-- Q4  如何看有 reaction 的对话比没有 reaction 的更加 active

--  这个是个开发的 follow up，自己定一个小 metric 然后再写成 sql 我是就简单的按

有无 reaction 分成两组然后看平均 message 的数量。。

--  这里面试官又问了这样的 metric 有什么 downside

select
    case when num_reaction = 0 then 'No' else 'Yes' end as type,
    avg(num_message) as avg_message
from
    (select
        case when senderid < receiverid then senderid else receiverid end as user_1,
        case when senderid < receiverid then receiverid else senderid end as user_2,
        sum(case when reaction_flag is 'Y' then 1 else 0 end) as num_reaction,
        count(message_thread) as num_message
    from conversations
```

```
    group by 1,2)s
group by 1
--------------------------------------------------------------------------------
table 1: connection
user1 | user 2
123 | 456
456 | 123
table 2: interaction
sender | receiver | action | date
123 | 456 | comment| 2019-01-01
-- question: find friends who haven't had interactions last year
select
    c.user1,
    c.user2
from
    (select
        case when user1<user2 then user1 else user2 end as user1,
        case when user1<user2 then user2 else user1 end as user2
from connection) c
left join (
    select
        case when sender<receiver then sender else receiver end as user1,
        case when sender<receiver then receiver else sender end as user2,
        action,
        date
    from interaction
    where year(date) = year(curdate()) - 1)i
on c.user1 = i.user1
and c.user2 = i.user2
where i.action is null

----------------------------------------10. Ads------------------------------
--https://www.1point3acres.com/bbs/thread-429404-1-1.html
ads: date|advertiser_id|ad_id|spend
user: date|user_id|ad_id|price

-- 1.求 average/total spend per advertiser today(不好意思忘记是 average 还是 total 了)

-- follow-up： after we get this table, what will the distribution look like?


-- | date | spend(sum(spend)/count(advertise)) |
```

```
select
    advertiser,
    sum(spend) as spend_per_day
from ads
where `date` = curdate()
group by 1
--2.how many advertisers have at least one conversion
-- I guess conversion means the ad_id exist in user table
select
    count(distinct a.advertiser_id)
from ads a
join user u
on a.ad_id = u.ad_id

--------------------------------------------------------------------------------
-- Ads click table
-- |ad_id | user_id | status: {'click', 'view', 'hide'} |
-- 1:用什么 metrics 去衡量 Ads peformance？ 写 SQL 找出 performnce 最好的 AD
-- I assume clikc + hide = view and I user CTR here
select
    ad_id
from
    (select
        ad_id,
        rank() over(order by sum(status = 'click')*1.00/sum(status = 'view') desc) as rk
    from Ads
    group by 1)s
where rk = 1

--2：如果人 click 其中一个 ad，写 SQL 推荐下一个 show 给这个 user 的 ad
-- the highest ctr ads except the clicked one
select
    ad_id
from
    (select
        ad_id,
        rank() over(order by sum(status = 'click')*1.00/sum(status = 'view') desc) as rk
    from Ads
    where ad_id not in (select ad_id from Ads where status = 'click' and user_id = 'xxx')
```

```
        group by 1)s
where rk = 1


-- 3:显示 user CTR 的 distribution

-- Questions: choose a bin size or just count frequency if exist? I'll try both ways
--a. just count frequency if exist
select
        CTR,
        count(ad_id) as frequency
from
        (select
                ad_id,
                round(ifnull(
                        sum(status = 'click')*100.00/nullif(sum(status = 'view'),0)
                        ,0),2) as CTR
        from Ads
        group by 1)s
group by 1

-- b. choose a bin size of 10
with recursive cte as (
        select 0 as CTR
        union all
        select CTR +10 as CTR
        from cte
        where CTR < 100
)

select
        c.CTR,
        count(ad_id) as frequency
from cte c
left join
        (select
                ad_id,
                round(ifnull(
                        sum(status = 'click')*100.00/nullif(sum(status = 'view'),0)
                        ,0),2) as CTR
        from Ads
        group by 1)s
```

```
on s.CTR - c.CTR >= 0
and s.CTR - c.CTR < 10
group by 1


-- 4:  还有什么别的 metrics that we should derive and monitor from this table.

-- hide rate by ads and user

--------------------------------------------------------------------------------
Ad4ad
Date            | event    | user_id | unit_id | ad_id |    cost | spend
2018-08-01 impression       123          1111         null       0.12       null
2018-08-01 impression       123          1111         null       0.15       null
2018-08-01 impression       123          1111         null       0.12       null
2018-08-01 impression       456          2222         null       0.14       null
2018-08-01 click               456          2222         null       null       null
2018-08-01 create_ad         456          2222         9988       null       10

User
User_id | country | age
123           German       38
456           China         20
789           US             28


-- Q1. last 30 days, by country, total spend (问的是 facebook 的 spend 就是表里的 cost)

of the product
select
      u.country,
      sum(cost) as sum_cost
from Ad a
join User u
on a.user_id = u.user_id
Where datediff(day, current_date(), date) <= 30
group by 1
-- Q2. how many impressions before users create an ad given a unit?

select
      unit_id,
      avg(n_impression) as avg_impression
from
```

```
    (select
        c.unit_id,
        c.ad_id,
        count(i.event) as n_impression
    from Ad c
    join Ad i
    on c.user_id = i.user_id
    and c.unit_id = i.unit_id
    and i.event = 'impression'
    where c.ad_id is not null
    group by 1,2)s
group by 1
```

-- Q3. 有个 ads_rolling table，是每个 ads 的 lifetime_spend 和 lifetime_revenue。问怎么把每 天新的信息加进去 update 这个 table.

ads_rolling: | unit_id | lifetime_spend | lifetime_revenue

```
select
    unit_id,
    sum(lifetime_spend) as lifetime_spend,
    sum(lifetime_revenue) as lifetime_revenue
from
(select
    *
from ads_rolling

union all

select
    unit_id,
    ifnull(sum(cost),0) as lifetime_spend,
    ifnull(sum(spend),0) as lifetime_revenue
from Ad
where date = Curdate()
group by 1)
group by 1
```

Ad_engagement table:
User_id, session_id, ad_id, action(view, click, hide), timestamp

Q1: find the ad that performs the best.
--clarify: multiple views/clicks
-- define best performance as highest ctr

```
select
    ad_id,
    sum(action = 'clicks')*100.00/ sum(actions = 'view' or actions = 'hide')
from Ad_engagement
group by 1
order by 2 desc
limit 1
```

-- Q2:given a user click ad_id=123, what's the next ad to show him. SQL

```
select
    ad_id,
    sum(action = 'clicks')*100.00/ sum(actions = 'view' or actions = 'hide')
from Ad_engagement
where ad_id <> '123'
group by 1
order by 2 desc
limit 1
```

Q3. Average time after people view a ad to click.

```
select
    avg(timestamediff(Second, click_time,view_time)) as time_gap
from
    (select
        v.user_id,
        v.ad_id,
        min(v.timestamp) as view_time,
        min(c.timestamp) as click_time
    from Ad_engagement v
    left join Ad_engagement c
    on v.user_id = c.user_id
    and v.ad_id = c.ad_id
    and c.action = 'click'
    where v.action = 'view'
    group by 1,2)s
```

---

advertiser: date | advertiser_id | status{'banned','no'} | spend

1.昨天百分之多少的正常 advertiser 是活跃的（有付钱)

```
select
    sum(spend >0 )*100.00/count(*) as active_percent
from advertiser
where status = 'no'
and datediff(Curdate(),date) = 1
```

2.昨天有多少被封了

```
--  可能不用 not in

select
    count(distinct advertiser_id) as banned_yesterday
from advertiser
where status = 'banned'
and advertiser_id not in
(select distinct
    advertiser_id
from advertise
where status = 'banned'
and datediff(Curdate(),date)>1)
```

3. 如果让 banned 账户申诉重启，怎么衡量经济利益

就给回复的账户一个独立的 status，比如 resurrected。然后每天看 resurrected 账户

贡献的利润所占总利润的百分比就行了，不用写 sql。

```
/*We are studying ecommerce advertisers on FB over a certain time period (say a week).
The time period does not matter for this problem.*/
adv_info:advertiser_id|| ad_id|| spend (primary: ad_id)
-- adv_info:Contains information on advertisers.
-- Advertiser_id is id of advertiser
-- ad_id is id of an ad being run by advertiser
-- spend is the amount of money in $ that advertiser pays Facebook for ad-id to show it to
FB users.

ad_info:ad_id|| user_id|| price (primary: ad_id, user_id)
-- price is how much the user_id spend through this ad.,assuming all prices > 0.
```

Q1: What would the average advertiser spend on Facebook? Your query should return a

single number.

```
select
    sum(spend)/count(distinct advertiser_id) as avg_spend
from adv_info
```

Q2. The fraction of advertisers has at least one conversion.

```
select
    count(distinct s.advertiser_id)/count(distinct a.advertiser_id) as adv_percent
from adv_info a
left join
    (select distinct
        advertiser_id
    from adv_info a
    join ad_info s
    on a.ad_id = s.ad_id)s
on a.advertiser_id = s.advertiser_id
```

----------11.Spam----------

```
-- Given the table about posts
User_actions: date | user_id | post_id | content_type | extra
-- content_type : 'view', 'comment', 'photo', 'report'
-- extra : post text, comment text, report type = {'SPAM', ...}

-- 1. how many posts were reported yesterday for each report reason?

select
    extra,
    count(distinct post_id) as report
from User_actions
where content_type = 'report'
group by 1
-- 2. Introduce a new table: reviewer_removals, please calculate what percent of daily content that users view on FB is actually spam?
-- reviewer_removals: ds(date, String) | reviewer_id |post_id
--Q: how do we define spam: report as spam and also removed
select
    a.date,
    count(distinct s.post_id)/count(distinct a.post_id)
from user_actions a
left join
    (select distinct
```

```
            post_id
        from user_actions u
        join reviewer_removals r
        on u.date = r.ds
        and u.post_id = r.post_id
        and content_type = 'report'
        and extra = 'SPAM')s
on a.post_id = s.post_id
where a.content_type = 'view'
group by 1

-- 3. How to find the user who abuses this spam system?
-- High spam report but percentage of spams actually removed by reviewer is low

select
        a.user_id,
        count(distinct a.post_id) as num_report
        coutn(distinct r.post_id)/count(distinct a.post_id) as percent_remove
from User_actions a
left join reviewer_removals r
on u.post_id = r.post_id
where content_type = 'report'
and extra = 'SPAM'
group by 1
order by 2 desc 3 asc

-- 4. What percent of yesterday's content views were on content reported for spam?
-- num: report as spam/view

select
        count(distinct r.post_id)/count(v.post_id)
from User_actions v
left join User_actions r
on v.post_id = r.post_id
and r.content_type = 'report'
and r.extra = 'SPAM'
where v.content_type = 'view'
and v.date = date_add(curdate(), interval -1 day)
```

Tech Analyis: Sql question: valid report 问题。 the table has the following fields:

report
date, owner_id, content_id, reporter_id, is_valid.
1: % of users who reported at least one valid report in December 2017?

-- by reporter(reporter 中有百分之多少的人至少有一个 valid report)

```
select
    count(distinct v.reporter_id)*1.00/ count(distinct r.reporter_id)
from report r
left join
    (select distinct
        reporter_id
    from report
    where is_valid = 1
    and year(date) = 2017
    and month(date) = 12)v
on r.reporter_id = v.reporter_id
where year(date) = 2017
and month(date) = 12
```

-- by owner (被 report 的人中有百分之多少的人有至少一个的 validate report)

```
select
    count(distinct r2.owner_id)/count(distinct r1.owner_id) as percent
from report r1
left join report r2
on r1.owner_id = r2.owner_id
and r2.is_valid = 1
and year(r2.date) = 2017
and month(r2.date) = 12
where year(r1.date) = 2017
and month(r1.date) = 12
```

2. find the users who are spamming the reporting tool?

第二个问题比较 confusing， 就是要 rank the reporter_id by counting the content_id they reported regardless the report is valid or not.

```
-- low valid rate
select
    reporter_id,
```

```
    sum(is_valid = 0)*1.00/count(*) as not_valid_rate
from report
group by 1
order by 2 desc
```

---------- 12.COMPOSER----------

-- FB is launching a 'composer' feature on the phone which helps user to post faster.

COMPOSER

user_id | date | event("enter", "post", "cancel")

User

user_id | date | country | dau_flag

dau_flag：active user ， 0 or 1

Q1: what success metric you will define for the feature? Write sql to calculate the success metric.

-- % of dau use it

```
select
    u.date,
    count(distinct c.user_id)/ count(distinct u.user_id) as daily_usage
from User u
left join composer c
on u.user_id = c.user_id
and u.date = c.date
where u.dau_flat = 1
group by 1
```

-- % of users successfully make a post

```
select
    count(distinct p.user_id)/
    count(distinct c.user_id) as percent_post
from composer c
left join
    (select distinct
        user_id
    from composer
    where event = 'post')p
on c.user_id = p.user)id
```

```
-- what is the average rate of successful post last week?
SELECT
        date,
        SUM(event = 'post')/ SUM(event = 'enter') as avg_success_rate
FROM Composer
WHERE datediff(day, date, current_date()) <= 7
GROUP BY 1
```

```
Q2: what is the average rate of post for daily active users by country on today?
select
        u.country,
        SUM(c.event = 'post')/ count(distinct u.user_id) as avg_success_rate
from user u
left join composer c
on u.user_id = c.user_id
where u.dau_flag = 1
and u.date = Curdate()
group by 1
```

----------13.Search----------

user_search 和 search_result

user_search

userid	date	queryid
10032	10122020	1
10032	10122020	2
10033	10132020	3

search_result

success 代表用户点击了，1 为点击，0 为没点击。

queryid	type	sucess
1	event	1
2	friends	0
3	event	0

1.求最近一周排名前 10 的搜索量最大的 user

```
select
    userid,
    count(queryid) as ct
from user_search
group by 1
order by 2
limit 10
```

2.求 对于 user 成功找到 event 的并且 sucess 的 user 比例

```
-- event success user/ event serch user
select
    (case when (type = 'event' and success = 1) then 1 else 0 end)*1.00/count(distinct
user_id) as u_percent
from
    (select distinct
        user_id,
        type,
        success
    from user_search s
    left join user_result r
    on s.queryid = r.queryid)s
```

Table: search_id, query, position, result_relevance

问题是 定义 metrics 来衡量 search quality

```
--leetcode 1211
search {search_id (int), query (varchar), position (int), result_relevance (int) [1-5] }
```

Position 表示 search 结果出现在页面的位置，相当于 rank，正整数，越小代表越靠前

result_relevance 代表 search 结果跟 query 的相关程度，范围 1 到 5， 越大代表越相关。比如：

1, “dog”, 1, 5
2, “dog”, 3, 4
3, “cat”, 2, 2

Q1: 定义一个 metric to measure search quality，既要考虑 position 也要考虑相关程度，计算每个 query 的 metric

```
--define search quality as result_relevance/position
select
    query,
    avg(result_relevance/position) as quality
from search
group by 1
```

Q2: 假如某个 query 的所有 relevance 都低于 3, 就定义为一个 quality 不好的 search，找到这些 query 在所有 query 中占的比例

```
select
    count(distinct s2.query)*100.00/count(distinct s1.query) as bad_query
from search s1
left join
    (select distinct
        query
    from search
    where query not in
        (select distinct
            query
        from search
        where result_relevance >=3))s2
on s1.query = s2.query
```

```
### Table 1: searches
### columns:-baidu 1point3acres
* date (string), date of the search
* search_id (int), the unique identifier of each search
* user_id (int), the unique identifier of the searcher
* age_group (string), ('U30', '30-50', '50+')
* search_query (string), the text of the search query
```

Sample Rows:

```
        date         | search_id         | user_id    | age_group | search_query
-------------------------------------------------------------------------------------------
'2020-01-01'  |       101       | 9991      |     'U30'    |    'michael jackson'
'2020-01-01'  |       102       | 9991      |    'U30'    |    'menlo park'
'2020-01-01'  |       103       | 5555      |    '30-50' |    'john'
'2020-01-01'  |       104       | 1234      |    '50+'    |    'funny dogs'
```

Table 2: search_results
columns:
* date (string), date of the search action
* search_id (int), the unique identifier of each search
* result_id (int), the unique identifier of the result
* result_type (string), ('page', 'event', 'group','person','post', etc.)
* clicked (boolean), did the user click on the result?

Sample Rows:

```
date                | search_id | result_id | result_type | clicked
-----------------------------------------------------------------------------------------
'2020-01-01' |       101       |       1001      |       'page'    |    TRUE
'2020-01-01' |       101       |       1002      |       'event'   |    FALSE
'2020-01-01' |       101       |       1003      |       'event'    |    FALSE
'2020-01-01' |       101       |       1004      |       'group'    |    FALSE
```

Q1: by each age group, how many unique users searched for "john" in the last 7 days?

```
select
      age_group,
      count(distinct user_id) as ct
from searches
where datediff(cur_date(), date) <= 7
and search_query = 'john'
group by 1
```

Q2: what are the top 10 search terms that are most likely to return at least one result about an Event?

```
select
      search_query,
      sum(result_type = 'event')*100.00/count(*) as percent_event
from searches s
join search_results r
```

on s.search_id = r.search_id
group by 1
order by 2 desc
limit 10
----------14. fundrasier----------

--
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=637243&extra=page%3D7%26filter%3Dsortid%26sortid%3D311%26searchoption%5B3046%5D%5Bvalue%5D%3D2%26searchoption%5B3046%5D%5Btype%5D%3Dradio%26sortid%3D311%26orderby%3Ddateline

user_fundraiser_summary
Columns:
date STRING (format - 2020-01-20)
user_id BIGINT (format - 81238123) - unique identifier for each user on FB
amount_donated BIGINT (format - 1000), amount in dollars
num_fundraiser_viewed INT (format - 1, 2, 3 etc) - total # of fundraisers viewed
num_fundraiser_donated INT (format - 1, 2, 3 etc) - total # of fundraisers donated
--Users don't need to donate to show up in this table but need to have viewed at least one fundraiser

all_fb_users: Daily snapshot of all FB users along with some demographic information and if they have been active in the last 30 days
Columns:
date STRING (format - 2020-01-20)
user_id BIGINT (format - 81238123) - unique identifier for each user on FB
country STRING (format - US, CA etc) - two letter country code. 1point3acres
locale (format - en_US, fr_FR etc) - language code
primary_interface STRING (format - desktop, ios, android, msite) - 4 possible values
30d_active STRING (format - yes or no)

Question 1: How many users donated more than $100 in the last 30 days?
```
select
	count(user_id) as ct
from
	(select
		user_id,
	from user_fundraiser_summary
	where datediff(curdate(), date) <= 30
```

```
    group by 1
    having sum(amount_donated) >100)s
```

Question 2: Broken down by locale, what is

a) the total amount of money donated in the last 30 days

```
-- assume user's demographic information does not change over day
select
    u.locale,
    sum(amount_donated) as sum_amount
from user_fundraiser_summary f
join all_fb_users u
on f.user_id = u.user_id
where datediff(curdate(), f.date) <= 30
group by 1
```

b) the percentage of 30d active users who viewed a fundraiser in the last 30 days

```
-- what if user's active status changed over the last 30 days?

select
    u.locale,
    count(distinct f.user_id)*100.00/count(distinct u.user_id) as percent_view
from all_fb_users u
left join user_fundraiser_summary f
on u.user_id = f.user_id
and u.date = f.date
and datediff(curdate(), f.date) <= 30
where datediff(curdate(), u.date) <= 30
and 30d_active = 'yes'
group by 1
```

c) the percentage of 30d active users who donated to fundraiser in the last 30 days

```
select
    u.locale,
    count(distinct f.user_id)*100.00/count(distinct u.user_id) as percent_view
from all_fb_users u
left join user_fundraiser_summary f
on u.user_id = f.user_id
and u.date = f.date
and datediff(curdate(), f.date) <= 30
and f.num_fundraiser_donated >= 1
where datediff(curdate(), u.date) <= 30
```

```
and 30d_active = 'yes'
group by 1
----------15. Video(watch)----------
Users {id|country},
Videos{userid|videoid|duration|...}
-- 1. find total_watch_time for each country
select
	country,
	sum(duration) as total_watch_time
from users u
join videos v
on u.id = v.userid
group by 1

-- 2. find top three countries with highest total watch time
-- limit or window(for tie)
select
	country
from
	(select
		country,
		row_number() over(partition by country order by sum(duration) desc) as rk
	from users u
	join videos v
	on u.id = v.userid
	group by 1)s
where rk <= 3

-- 3. find average of watch time for non-top 3 countries (answer is just one number)
select
	avg(duration) as total_watch_time
from users u
join videos v
on u.id = v.userid
where country not in
	(select
		country
	from
		(select
			country,
			row_number() over(partition by country order by sum(duration) desc) as
```

```
rk
            from users u
            join videos v
            on u.id = v.userid
            group by 1)s
      where rk <= 3)
group by 1
```

----------16.feature----------

There is a table that tracks every time a user turns a feature on or off, with columns
table: user_id |action ("on" or "off") |date | time

```
-- 1) How many users turned the feature on today?
-- assuem: last status is on
select
      sum(action = 'on') as ct_user
from
(select
      user_id,
      action,
      rank() over(partition by user_id order by time desc) as rk
from table
where date = curdate())s
where rk = 1
-- How many users have ever turned the feature on?
select
      count(distinct user_id) as ct
from table
where action = "on"

-- 2) Create a table that tracks the user last status every day.
select
      date,
      user_id,
      action
from
(select
      user_id,
      date,
      action,
      rank() over(partition by user_id, date order by time desc) as rk
from table)s
```

```
where rk = 1

-- 3) In a table that tracks the status of every user every day, how would you add today's data to it?

select *
from everyday_status

union all

select
	e.user_id,
	cur_date() as date,
	ifnull(c.action, e.action) as action
from everyday_status e
left join
	(select
		user_id,
		action,
		rank() over(partition by user_id order by time desc) as rk
	from table
	where date = curdate())c
on e.user_id = c.user_id
and c.rk = 1
where e.date = date_add(curdate(),-1 interval day)
```

4) 如何找出在一天之内始终保持 feature on 的人

```
-- use not in
-----------------------------------------17. Other-----------------------------------------------
```

根据 cus_id 和 date 生成 count，count 的意思是过去 7 天（不包含当天），有几个不同的 cus_id 出现

```
cus_id | date | count
    c1   | 12/1 |     0
    c1   | 12/2 |     1
    c2   | 12/3 |     1
    c3   | 12/9 |     2

select
```

```
        t1.date,
        count(distinct t2.cus_id) as ct
from table t1
left join table t2
on datediff(t1.date, t2.date) between 1 and 7
group by 1
```
