SQL- Intermediate - Part 3

Challenges

https://www.hackerrank.com/challenges/challenges/problem?isFullScreen=true

hacker_id, name and total no of challenges created by each student

sort it by descending order.
if more than one
sort it by hacker_id

if count is less than max. no of challenges exclude them iff more than one student created same no of challenges.

Hackers :-	Challenges
hacker_id	challenge_id
name	hacker_id.

Challenge to write max_challenge count

maximum of the challenges = 50

```
SELECT MAX (challenges_count) FROM
(SELECT count (challenge_id) as challenges_count
FROM challenges GROUP BY hacker_id)
```

if it is maximum duplicates are allowed from question

gives count of no. of. challenges created by each hacker

Now the next part is how to find the number of challenges which are duplicate which means that there count ≥ 1.

```
SELECT challenges_count FROM
( SELECT challenges_count from
(SELECT count (challenge_id) as challenge_count
FROM challenges
GROUP BY hacker_id) as Subquery
GROUP BY challenges_count
HAVING COUNT (*) > 1))
```

group by along challenges to check there count

list of challenges which are duplicates

```
SELECT h.hacker_id, h.name, COUNT(c.challenge_id) as challenges_count
FROM Hackers h
JOIN Challenges c ON h.hacker_id = c.hacker_id
GROUP BY h.hacker_id, h.name
HAVING COUNT(c.challenge_id) = (SELECT MAX(challenges_count) FROM
                                (SELECT COUNT(challenge_id) as challenges_count
                                FROM Challenges
                                GROUP BY hacker_id) as SubQuery)
   OR (COUNT(c.challenge_id) NOT IN (SELECT challenges_count FROM
                                    (SELECT COUNT(challenge_id) as challenges_count
                                    FROM Challenges
                                    GROUP BY hacker_id) as SubQuery
                                    GROUP BY challenges_count
                                    HAVING COUNT(*) > 1))
ORDER BY challenges_count DESC, h.hacker_id;
```

join on the table

↳ maximum duplicates allowed

check for duplicates and don't consider them

challenges_count in descending order

hacker_id in ascending order.

Symmetric Pairs

https://www.hackerrank.com/challenges/symmetric-pairs/problem?isFullScreen=true

Use of CTE in SQL → named result set in query

Syntax :- WITH cte as (select statement query.)

For example in this case

with a as (select x, y, row_number() over() as r from functions)

Condn in this question:-

Symmetric pairs $x_1 = y_2$ and $x_2 = y_1$

Query by ascending order of x_1 and order s.t $x_1 \leq y_1$

```
with a as ( select x, y, row_number()  over() r from functions ),
b as ( select x, y, row_number()  over() r from functions )

select distinct a.x,a.y
from a,b
where a.x=b.y and a.y=b.x and a.r!=b.r and a.x<=a.y
order by a.x;
```

self join

no repitition

according to question

order by x_1

Interviews

https://www.hackerrank.com/challenges/interviews/problem?isFullScreen=true

contest_id, hacker_id, name

Sums of total_submissions, total_accepted_submissions

total_views and total_unique_views

for each contest by contest_id

Exclude context if all sum = 0

Specific context → many college

1 college - 1 contest

Column	Type
contest_id	Integer
hacker_id	Integer
name	String

Contests

Column	Type
challenge_id	Integer
college_id	Integer

Colleges

Column	Type
challenge_id	Integer
total_views	Integer
total_unique_views	Integer

Challenges

Column	Type
challenge_id	Integer
total_views	Integer
total_unique_views	Integer

View_Stats

Column	Type
challenge_id	Integer
total_views	Integer
total_unique_views	Integer

Submission_Stats

```sql
SELECT
    cc.contest_id,
    cc.hacker_id,
    cc.name,
    COALESCE(SUM(sa.total_submissions_ch), 0) AS total_submission_contest,
    COALESCE(SUM(sa.total_accepted_submissions_ch), 0) AS
total_accepted_submission_contest,
    COALESCE(SUM(va.total_views_ch), 0) AS total_views_contest,
    COALESCE(SUM(va.total_unique_views_ch), 0) AS total_unique_views_contest
FROM
    (SELECT c.contest_id, c.hacker_id, c.name, ch.challenge_id
     FROM colleges col
     JOIN contests c ON col.contest_id = c.contest_id
     JOIN challenges ch ON ch.college_id = col.college_id) AS cc
LEFT JOIN
    (SELECT challenge_id,
            SUM(total_submissions) AS total_submissions_ch,
            SUM(total_accepted_submissions) AS total_accepted_submissions_ch
     FROM submission_stats
     GROUP BY challenge_id) AS sa ON sa.challenge_id = cc.challenge_id
LEFT JOIN
    (SELECT challenge_id,
            SUM(total_views) AS total_views_ch,
            SUM(total_unique_views) AS total_unique_views_ch
     FROM view_stats
     GROUP BY challenge_id) AS va ON va.challenge_id = cc.challenge_id
GROUP BY cc.contest_id, cc.hacker_id, cc.name
HAVING (total_submission_contest + total_accepted_submission_contest +
total_views_contest + total_unique_views_contest) > 0
ORDER BY cc.contest_id;
```

Since we are doing left join here we need to handle null values.

↳ table of contest and challenge joined with college since she forgot about which challenge are there in contest

↓ Sum over challenges

Submissions

↓ Sum over challenges

Views

why left join? → because there might or might not be data which is useful in the tables of views of submissions ↳ can contain nulls

But all the data of challenges, contest and colleges must be there left join