WITH stats\_table AS (

SELECT

cc.challenge\_id, cc.college\_id,SUM(vt.total\_views) AS ttotal\_views,SUM(vt.total\_unique\_views) AS ttotal\_unique\_views,SUM(st.total\_submissions) AS ttotal\_submissions,SUM(st.total\_accepted\_admissions) AS ttotal\_accepted\_admissions,

CASE

WHEN SUM(vt.total\_views) = 0 AND SUM(vt.total\_unique\_views) = 0 AND SUM(st.total\_submissions) AND SUM(st.total\_accepted\_admissions) = 0

THEN 0

ELSE 'Usable'

END AS criteria,

FROM

chall\_col AS cc

LEFT JOIN

views\_table AS vt

ON cc.challenge\_id = vt.challenge\_id

LEFT JOIN

submission\_table AS st

ON cc.challenge\_id = st.challenge\_id

WHERE criteria = 'Usable'

GROUP BY cc.challenge\_id)

SELECT concat(ha.contest\_id,' ',ha.hacker\_id,' ',ha.name,' ',ct.ttotal\_submissions,' ',ct.ttotal\_accepted\_admissions,' ',ct.ttotal\_views,' ',ct.ttotal\_unique\_views)

FROM

hacker AS ha

LEFT JOIN

(SELECT cco.college\_id,SUM(stats.ttotal\_views) AS tttotal\_views,SUM(stats.ttotal\_unique\_views) AS tttotal\_unique\_views,SUM(stats.ttotal\_submissions) AS tttotal\_submissions,SUM(stats.ttotal\_accepted\_admissions) AS tttotal\_accepted\_admissions

FROM

col\_con AS cco

LEFT JOIN

stats\_table AS stats

ON cco.college\_id = stats.college\_id

GROUP BY cco.college\_id) AS ct

ON ha.contest\_id = ct.contest\_id

ORDER BY ha.contest\_id

select

contests.contest\_id,

contests.hacker\_id,

contests.name,

sum(submissions\_sums.sum\_submissions),

sum(submissions\_sums.sum\_accepted\_submissions),

sum(views\_sums.sum\_views),

sum(views\_sums.sum\_unique\_views)

from contests

join colleges on contests.contest\_id = colleges.contest\_id

join challenges on colleges.college\_id = challenges.college\_id

-- subquery to get total sums for the Submission stats.

-- these subqueries use left joins, so that the unrelated/empty information is not joined.

left join

(select

challenge\_id,

sum(total\_submissions) as sum\_submissions,

sum(total\_accepted\_submissions) as sum\_accepted\_submissions

from submission\_stats group by challenge\_id)

as submissions\_sums

on challenges.challenge\_id = submissions\_sums.challenge\_id

-- another subquery to get total sums for Views stats

left join

(select

challenge\_id,

sum(total\_views) as sum\_views,

sum(total\_unique\_views) as sum\_unique\_views

from view\_stats group by challenge\_id)

as views\_sums

on challenges.challenge\_id = views\_sums.challenge\_id

-- group the information per-contest so that everything is aggregated.

group by contests.contest\_id, contests.hacker\_id, contests.name

-- HAVING works like WHERE, except over aggregations, which is what we want here.

having (

sum(submissions\_sums.sum\_submissions) +

sum(submissions\_sums.sum\_accepted\_submissions) +

sum(views\_sums.sum\_views) +

sum(views\_sums.sum\_unique\_views)

) > 0

order by contests.contest\_id