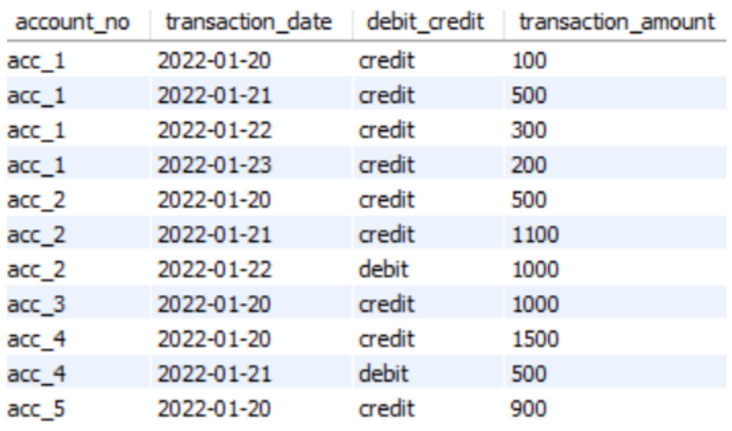
Applying SQL to solve Problems using Window Functions including Hacker rank interview Questions.

Question 1: Write a query to return the account number and transaction date when the account balance is reached to 1000. Please include only those accounts whose balance currently is >=1000.



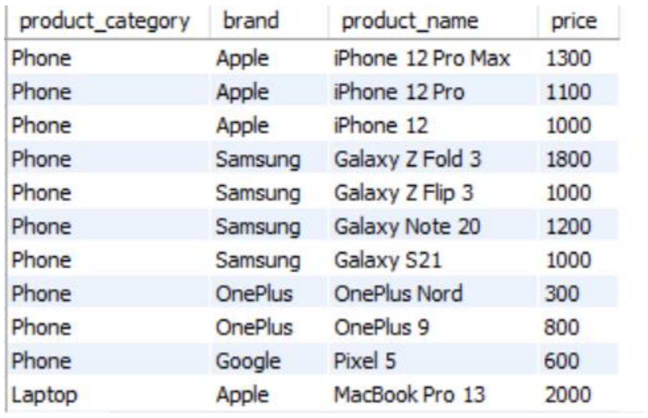
account_no	transaction_date	debit_credit	transaction_amount
acc_1	2022-01-20	credit	100
acc_1	2022-01-21	credit	500
acc_1	2022-01-22	credit	300
acc_1	2022-01-23	credit	200
acc_2	2022-01-20	credit	500
acc_2	2022-01-21	credit	1100
acc_2	2022-01-22	debit	1000
acc_3	2022-01-20	credit	1000
acc_4	2022-01-20	credit	1500
acc_4	2022-01-21	debit	500
acc_5	2022-01-20	credit	900

Solution:

```
with cte as (select account_no,transaction_date,
      case when debit_credit='credit'
      then transaction_amount
      else transaction_amount * -1
      end as balance
 from account_balance),
  cte1 as (select *, sum(balance) over (partition by account_no order by
transaction_date
                      range between unbounded preceding and unbounded following) as
final_balance,
                      sum(balance) over (partition by account_no order by transaction_date
                     ) as current_balance
 from cte)
 select account_no,min(transaction_date) as transaction_date  from cte1
 where final_balance>=1000 and current_balance>=1000
```

```
group by account_no
```

Question 2: Write a query to display the most expensive product under each category (corresponding to each record).



product_category	brand	product_name	price
Phone	Apple	iPhone 12 Pro Max	1300
Phone	Apple	iPhone 12 Pro	1100
Phone	Apple	iPhone 12	1000
Phone	Samsung	Galaxy Z Fold 3	1800
Phone	Samsung	Galaxy Z Flip 3	1000
Phone	Samsung	Galaxy Note 20	1200
Phone	Samsung	Galaxy S21	1000
Phone	OnePlus	OnePlus Nord	300
Phone	OnePlus	OnePlus 9	800
Phone	Google	Pixel 5	600
Laptop	Apple	MacBook Pro 13	2000

Solution 1:

```
select p1.product_name from(

select *, row_number() over(partition by product_category  order by price desc) as rn

from product) as p1

where p1.rn=1
```

Solution 2:

```
select p.product_category,p.product_name from(

select *,

first_value(product_name) over w as most_expensive ,

row_number() over w as rn

from product

window w as (partition by product_category order by price desc) ) p

where p.rn=1
```

Question 3: Write a query to display the second most expensive product on each category?

Solution 1:

```
select p1.product_name from (

   select *, row_number() over w as rn
from product
window w as (partition by product_category  order by price desc)) p1
where p1.rn=2
```

Solution 2:

```
select *,
nth_value(product_name, 2) over w as second_most_exp_product
from product
window w as (partition by product_category order by price desc
        range between unbounded preceding and unbounded following);
```

Question 4: Write a query to segregate all the expensive phones, mid range phones and the cheaper ones.

<u>Using NTILE of 3 segregation</u>

Solution: select p.product_name,case when p.segregation=1 then 'expensive phones'

```
                    when p.segregation=2 then 'mid range phones'
                    else 'cheaper phones' end as catagory from
            (select *,
            ntile(3) over(order by price desc) as segregation
            from product where product_category='Phone') p
```

Question 5: Query to fetch all products which are constituting the first 30% of the data in products table based on price.

Solution: /* Formula = Current Row no / Total no of rows */. When it gets the duplicate records (using the order by clause which is the price), it considers the **last value** of the record.

```
select *
from (
   select *,
   cume_dist() over (order by price desc)  as cumlative_distirbution,
   round(cume_dist() over (order by price desc) *100,2)  as cume_dist_percetage
   from product) x
```

```
where x.cume_dist_percetage <= 30;
```

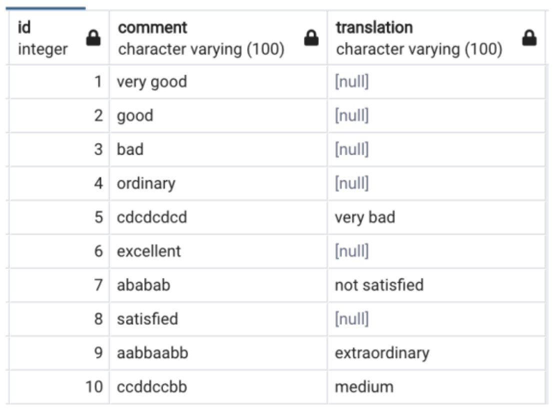
Question 6: Query to identify how much percentage more expensive is "Galaxy Z Fold 3" when compared to all products.

Solution: Using PERCENT_RANK (relative rank of the current row / Percentage Ranking).

```
/* Formula = Current Row No - 1 / Total no of rows - 1 */

    select product_name, percentage

    from (

        select *,

        percent_rank() over(order by price ) ,

        round(percent_rank() over(order by price ) * 100, 2) as percentage

        from product) x

    where x.product_name='Galaxy Z Fold 3';
```

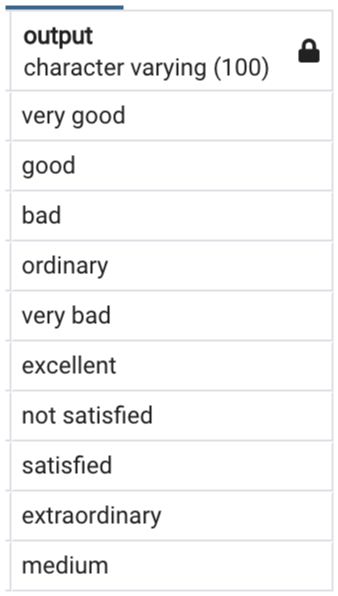
Question 7: Write an SQL query to display the correct message (meaningful message) from the input

Input:



id integer	comment character varying (100)	translation character varying (100)
1	very good	[null]
2	good	[null]
3	bad	[null]
4	ordinary	[null]
5	cdcdcdcd	very bad
6	excellent	[null]
7	ababab	not satisfied
8	satisfied	[null]
9	aabbaabb	extraordinary
10	ccddccbb	medium

Expected Output:

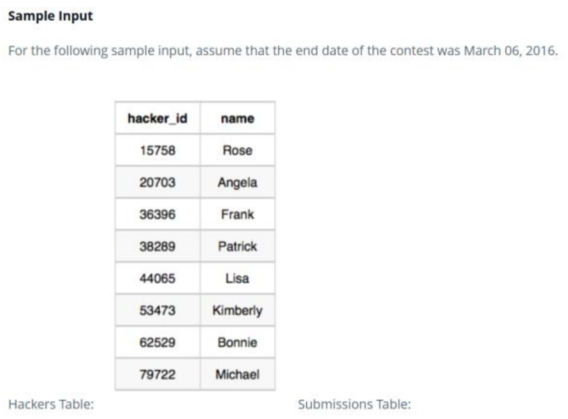


output character varying (100)
very good
good
bad
ordinary
very bad
excellent
not satisfied
satisfied
extraordinary
medium

Solution:

```
select
case
when translation is NULL then comment
else translation
end as output
from comments_and_translations;
```

Question 8: Julia conducted a 15 days of learning SQL contest. The start date of the contest was *March 01, 2016* and the end date was *March 15, 2016*. Write a query to print total number of unique hackers who made at least submission each day (starting on the first day of the contest), and find the *hacker_id* and *name* of the hacker who made maximum number of submissions each day. If more than one such hacker has a maximum number of submissions, print the lowest *hacker_id*. The query should print this information for each day of the contest, sorted by the date.

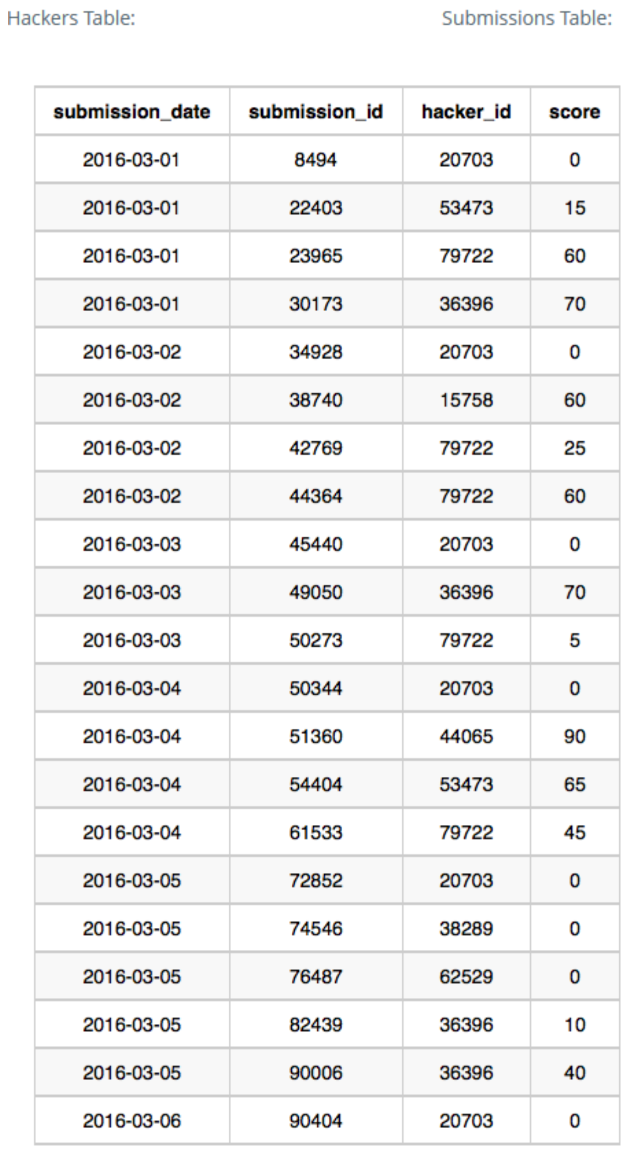


Sample Input

For the following sample input, assume that the end date of the contest was March 06, 2016.

hacker_id	name
15758	Rose
20703	Angela
36396	Frank
38289	Patrick
44065	Lisa
53473	Kimberly
62529	Bonnie
79722	Michael

Hackers Table: Submissions Table:



Hackers Table: Submissions Table:

submission_date	submission_id	hacker_id	score
2016-03-01	8494	20703	0
2016-03-01	22403	53473	15
2016-03-01	23965	79722	60
2016-03-01	30173	36396	70
2016-03-02	34928	20703	0
2016-03-02	38740	15758	60
2016-03-02	42769	79722	25
2016-03-02	44364	79722	60
2016-03-03	45440	20703	0
2016-03-03	49050	36396	70
2016-03-03	50273	79722	5
2016-03-04	50344	20703	0
2016-03-04	51360	44065	90
2016-03-04	54404	53473	65
2016-03-04	61533	79722	45
2016-03-05	72852	20703	0
2016-03-05	74546	38289	0
2016-03-05	76487	62529	0
2016-03-05	82439	36396	10
2016-03-05	90006	36396	40
2016-03-06	90404	20703	0

Solution:

- Using recursive CTE.

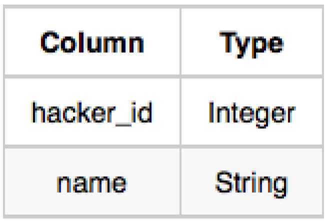
```
with recursive  hacker_submission_eachday as (
     select submission_date,hacker_id
     from submissions where submission_date =(select min(submission_date) from submissions)
     union
     select s.submission_date,s.hacker_id
     from hacker_submission_eachday h,submissions s
     where h.hacker_id=s.hacker_id and s.submission_date =date_add(h.submission_date,interval 1
day)
     and s.submission_date < date_add((select max(submission_date) from submissions) ,interval 1
day)
),
recursion_result as (select *
from hacker_submission_eachday),
unique_hackers as (select submission_date,count(distinct hacker_id) as no_unique_hacker
from recursion_result
group by submission_date),
cte as (select s.submission_date,hh.hacker_id,hh.name,count(hh.hacker_id) as Count_Submission
from submissions s,hackers hh
where s.hacker_id=hh.hacker_id
group by 1,2
),
hacker_id_name_max_submission as (select * ,
row_number() over(partition by submission_date order by Count_Submission desc) as rn
from cte ),
final_result_hacker_id_name as (select submission_date,hacker_id,name
from hacker_id_name_max_submission where rn=1)
select h1.submission_date,h1.no_unique_hacker,h2.hacker_id,h2.name
from unique_hackers h1,final_result_hacker_id_name h2
where h1.submission_date=h2.submission_date
```

Question 9: Julia asked her students to create some coding challenges. Write a query to print the *hacker_id*, *name*, and the total number of challenges created by each student. Sort your results by the total number of challenges in descending order. If more than one student created the same number of challenges, then sort the result by *hacker_id*. If more than one student created the same number of challenges and the count is less than the maximum number of challenges created, then exclude those students from the result.

Input Format

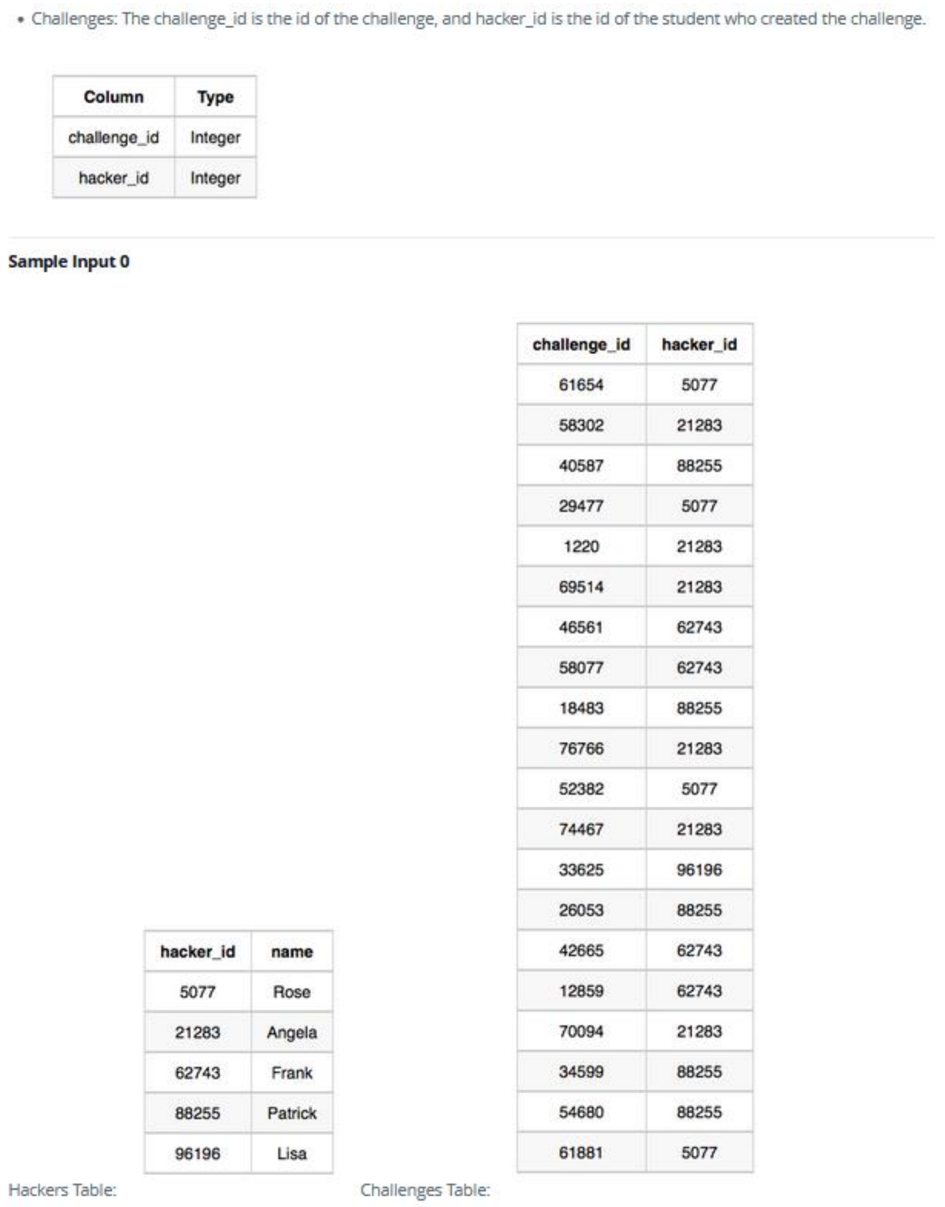
The following tables contain challenge data:

- *Hackers:* The *hacker_id* is the id of the hacker, and *name* is the name of the hacker.



Column	Type
hacker_id	Integer
name	String

- *Challenges:* The *challenge_id* is the id of the challenge, and *hacker_id* is the id of the student who created the challenge.



- Challenges: The challenge_id is the id of the challenge, and hacker_id is the id of the student who created the challenge.

Column	Type
challenge_id	Integer
hacker_id	Integer

Sample Input 0

Hackers Table:

hacker_id	name
5077	Rose
21283	Angela
62743	Frank
88255	Patrick
96196	Lisa

Challenges Table:

challenge_id	hacker_id
61654	5077
58302	21283
40587	88255
29477	5077
1220	21283
69514	21283
46561	62743
58077	62743
18483	88255
76766	21283
52382	5077
74467	21283
33625	96196
26053	88255
42665	62743
12859	62743
70094	21283
34599	88255
54680	88255
61881	5077

Solution:

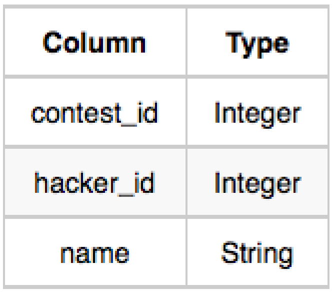
```
with hacker_challenge as (select h.hacker_id,h.name,count(c.challenge_id) as no_challenge
        from Challenges c , hackers h
        where c.hacker_id=h.hacker_id
        group by c.hacker_id
        order by count(c.challenge_id) desc),
        max_challenges as (select max(no_challenge) as max_no_challenge from hacker_challenge),
        exclude_hackers as (select h11.hacker_id,h11.name,h11.no_challenge
        from hacker_challenge h11,hacker_challenge h22
        where h11.no_challenge=h22.no_challenge and
            h11.hacker_id<>h22.hacker_id and h11.no_challenge<(select max_no_challenge from
        max_challenges)
            order by h11.hacker_id desc)
        select *
        from hacker_challenge where hacker_id not in (select hacker_id from exclude_hackers )
```

Question 10: Samantha interviews many candidates from different colleges using coding challenges and contests. Write a query to print the *contest_id*, *hacker_id*, *name*, and the sums of *total_submissions*, *total_accepted_submissions*, *total_views*, and *total_unique_views* for each contest sorted by *contest_id*. Exclude the contest from the result if all four sums are 0

Note: A specific contest can be used to screen candidates at more than one college, but each college only holds 1 screening contest.

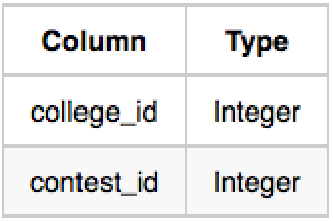
Input Format: The following tables hold interview data:

- *Contests:* The *contest_id* is the id of the contest, *hacker_id* is the id of the hacker who created the contest, and *name* is the name of the hacker.



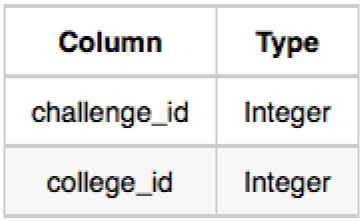
Column	Type
contest_id	Integer
hacker_id	Integer
name	String

- *Colleges:* The *college_id* is the id of the college, and *contest_id* is the id of the contest that Samantha used to screen the candidates.



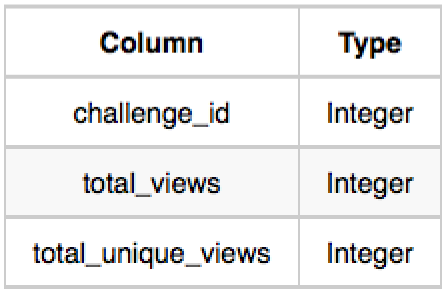
Column	Type
college_id	Integer
contest_id	Integer

- *Challenges:* The *challenge_id* is the id of the challenge that belongs to one of the contests whose contest_id Samantha forgot, and *college_id* is the id of the college where the challenge was given to candidates.



Column	Type
challenge_id	Integer
college_id	Integer

- *View_Stats:* The *challenge_id* is the id of the challenge, *total_views* is the number of times the challenge was viewed by candidates, and *total_unique_views* is the number of times the challenge was viewed by unique candidates.



Column	Type
challenge_id	Integer
total_views	Integer
total_unique_views	Integer

- *Submission_Stats:* The *challenge_id* is the id of the challenge, *total_submissions* is the number of submissions for the challenge, and *total_accepted_submission* is the number of submissions that achieved full scores.

Column	Type
challenge_id	Integer
total_submissions	Integer
total_accepted_submissions	Integer

Sample Input

Contests Table:

contest_id	hacker_id	name
66406	17973	Rose
66556	79153	Angela
94828	80275	Frank

Colleges Table:

college_id	contest_id
11219	66406
32473	66556
56685	94828

Challenges Table:

challenge_id	college_id
18765	11219
47127	11219
60292	32473
72974	56685

View_Stats Table:

challenge_id	total_views	total_unique_views
47127	26	19
47127	15	14
18765	43	10
18765	72	13
75516	35	17
60292	11	10
72974	41	15
75516	75	11

Submission_Stats Table:

challenge_id	total_submissions	total_accepted_submissions
75516	34	12
47127	27	10
47127	56	18
75516	74	12
75516	83	8
72974	68	24
72974	82	14
47127	28	11

Sample Output

```
66406 17973 Rose 111 39 156 56
66556 79153 Angela 0 0 11 10
94828 80275 Frank 150 38 41 15
```

Solution:

```
with contests_with_college as (select c1.contest_id,c1.hacker_id,c1.name,c2.college_id
        from contest as c1,college as c2
        where c1.contest_id=c2.contest_id),
        contest_with_challenge as (
        select ch1.contest_id, ch1.hacker_id,ch1.name,ch.challenge_id, ch1.college_id
        from contests_with_college as ch1 ,challenges as ch
        where ch.college_id=ch1.college_id),
        contest_submissions as (select cc.contest_id,cc.hacker_id,cc.name,
                                                    cc.challenge_id, s.total_submissions ,
                                                    s.total_accepted_submissions
        from submission_stats s,contest_with_challenge as cc
        where s.challenge_id=cc.challenge_id),
        contest_submissions_result as (select contest_id,hacker_id,name,
                                                    challenge_id, sum(total_submissions) as
        total_submissions ,
                                                    sum(total_accepted_submissions) as
        total_accepted_submissions
        from contest_submissions
        group by 1,2,3,4),
        contest_statistics as (select   cc.contest_id, cc.hacker_id,cc.name, cc.college_id,
            sum(vs.total_views) as toal_view,sum(vs.total_unique_views) as total_unique_views
        from contest_with_challenge as cc,view_stats as vs
        where cc.challenge_id=vs.challenge_id
        group by 1,2,3)
        select   css.contest_id, css.hacker_id, css.name,css1.total_submissions,
        css1.total_accepted_submissions,css.toal_view,css.total_unique_views
        from contest_statistics css left join contest_submissions_result css1
        on css.contest_id=css1.contest_id
```