**Experiment 1.4**

**Student Name:   Gaurav Kumar                                UID: 22MCC20177**

**Branch:   CC-DevOps                                                        Section/Group: 1/B**

**Semester:   I                                                               Date of Performance: 05/10/2022**

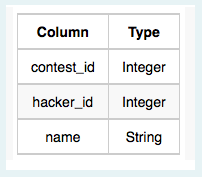
**Subject Name: ADBMS                     Subject Code: 22CAP-647**

1. **Task to be done:**

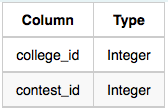
Samantha interviews many candidates from different colleges using coding challenges and contests. Write a query to print the contest\_id, hacker\_id, name, and the sums of total\_submissions, total\_accepted\_submissions, total\_views, and total\_unique\_views for each contest sorted by contest\_id. 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 screening contest.

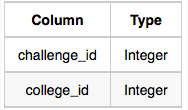
**Input Format**

The following tables hold interview data:

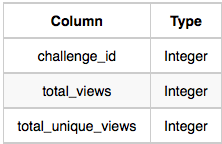
* **Contests:** The contest\_id is the id of the contest, hacker\_id is the id of the hacker who created the contest, and name is the name of the hacker.



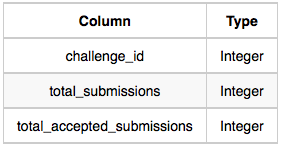
* **Colleges:** The college\_id is the id of the college, and contest\_id is the id of the contest that Samantha used to screen the candidates.



* **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.



* **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.



* **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.

**Implement the above experiment**

**mysql>** create database WS\_1\_4;

**Query OK, 1 row affected (0.00 sec)**

**mysql>** use ws\_1\_4

**Database changed**

**mysql>** create table contests(

**->** contest\_id int primary key,

**->** hacker\_id int unique not null,

**->** name varchar(30));

**Query OK, 0 rows affected (0.04 sec)**

**mysql>** create table colleges(

**->** college\_id int primary key,

**->**

**->** contest\_id int references contests(contest\_id));

**Query OK, 0 rows affected (0.01 sec)**

**mysql>** create table challenges(

**->** challenge\_id int primary key,

**->** college\_id int references colleges(college\_id));

**Query OK, 0 rows affected (0.02 sec)**

**mysql>** create table view\_stats(

**->** challenge\_id int references challenges(challenge\_id),

**->** total\_views int,

**->** total\_unique\_view int);

**Query OK, 0 rows affected (0.02 sec)**

**mysql>** create table submission\_stats(

**->** challenge\_id int references challenges(challenge\_id),

**->** total\_submissions int,

**->**

**->** total\_accepted\_submissions int);

**Query OK, 0 rows affected (0.02 sec)**

**mysql>** insert into contests values(66406,17973,"Rose"),

**->** (66556,79153,"Angela"),

**->** (94828,80275,"Frank");

**Query OK, 3 rows affected (0.01 sec)**

**Records: 3 Duplicates: 0 Warnings: 0**

**mysql>** insert into colleges values

**->** (11219,66406),

**->** (32473,66556),

**->** (56685,94828);

**Query OK, 3 rows affected (0.01 sec)**

**Records: 3 Duplicates: 0 Warnings: 0**

**mysql>** insert into challenges values

**->** (18765,11219),

**->** (47127,11219),

**->** (60292,32473),

**->** (72974,56685);

**Query OK, 4 rows affected (0.01 sec)**

**Records: 4 Duplicates: 0 Warnings: 0**

**mysql>** insert into submission\_stats values

**->** (75516,34,12)

**->** ,(47127,27,10),

**->** (47127,56,18),

**->** (75516,74,12),

**->** (75516,83,8),

**->** (72974,68,24),

**->** (72974,82,14),

**->** (47127,28,11);

**Query OK, 8 rows affected (0.01 sec)**

**Records: 8 Duplicates: 0 Warnings: 0**

**mysql>** insert into view\_stats values

**->** (47127,26,19),

**->** (47127,15,14),

**->** (18765,43,10),

**->** (18765,72,13),

**->** (75516,35,17),

**->** (60292,11,10),

**->** (72974,41,15),

**->** (75516,75,11);

**Query OK, 8 rows affected (0.01 sec)**

**Records: 8 Duplicates: 0 Warnings: 0**

**Query :**

select

contests.contest\_id,

contests.hacker\_id,

contests.name,

sum(submissions\_sums.sum\_submissions) as submissionSum,

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

sum(views\_sums.sum\_views) as viewSum,

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

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\_view) 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) +

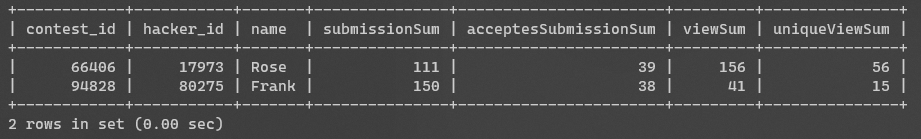
sum(views\_sums.sum\_views) +

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

) > 0

order by contests.contest\_id;

**Output :**



1. **Learning outcomes (What I have learnt):** 
   * 1. **Learn about Aggregate function**
     2. **Learn about group by clause**
     3. **Learn about having clause with aggregate functions**

**Evaluation Grid:**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Demonstration and Performance  (Quiz) |  | 22 |
| 2. | Worksheet |  | 8 |