**Q33:** Given the CITY and COUNTRY tables, query the names of all the continents (*COUNTRY.Continent*) and their respective average city populations (*CITY.Population*) rounded *down* to the nearest integer.

Note: *CITY.CountryCode* and *COUNTRY.Code* are matching key columns.

Input Format. The CITY and COUNTRY tables are described as follows:

****

****

**Solution:**

SELECT co.continent, FLOOR(AVG(ci.population))

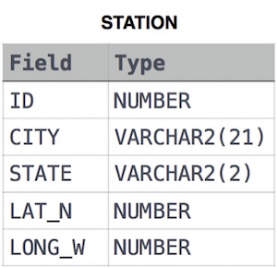
FROM CITY ci

JOIN COUNTRY co

ON co.code = ci.countrycode

GROUP BY co.continent;

**Q34:** Query the two cities in STATION with the shortest and longest *CITY* names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.  
The STATION table is described as follows:



where LAT\_N is the northern latitude and LONG\_W is the western longitude.

Sample Input: For example, CITY has four entries: DEF, ABC, PQRS and WXY.

Sample Output

ABC 3

PQRS 4

**Explanation**

When ordered alphabetically, the **CITY** names are listed as **ABC, DEF, PQRS,** and **WXY**, with lengths 3,3,4, and 3. The longest name is **PQRS**, but there are 3 options for shortest named city. Choose **ABC**, because it comes first alphabetically.

**Note**  
You can write two separate queries to get the desired output. It need not be a single query.

**Solution:**

SELECT CITY, LENGTH(CITY) FROM STATION ORDER BY LENGTH(CITY), CITY LIMIT 1;

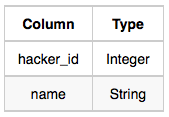
SELECT CITY, LENGTH(CITY) FROM STATION ORDER BY LENGTH(CITY) DESC, CITY LIMIT 1;

**Q35:** Julia just finished conducting a coding contest, and she needs your help assembling the leaderboard! Write a query to print the respective *hacker\_id* and *name* of hackers who achieved full scores for *more than one* challenge. Order your output in descending order by the total number of challenges in which the hacker earned a full score. If more than one hacker received full scores in same number of challenges, then sort them by ascending *hacker\_id*.

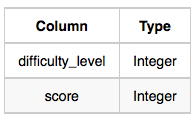
Input Format

The following tables contain contest data:

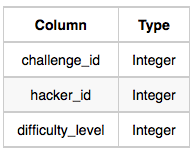
* *Hackers:* The *hacker\_id* is the id of the hacker, and *name* is the name of the hacker.



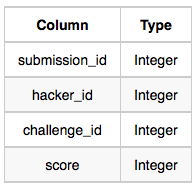
* *Difficulty:* The *difficult\_level* is the level of difficulty of the challenge, and *score* is the maximum score that can be achieved for a challenge at that difficulty level.



* *Challenges:* The *challenge\_id* is the id of the challenge, the *hacker\_id* is the id of the hacker who created the challenge, and *difficulty\_level* is the level of difficulty of the challenge.

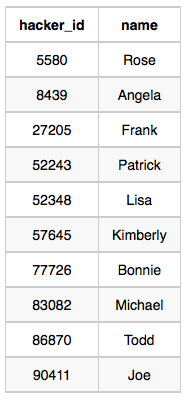


* *Submissions:* The *submission\_id* is the id of the submission, *hacker\_id* is the id of the hacker who made the submission, *challenge\_id* is the id of the challenge that the submission belongs to, and *score* is the score of the submission.

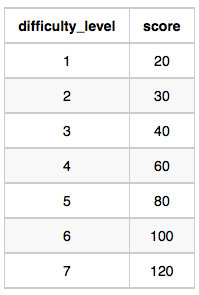


Sample Input

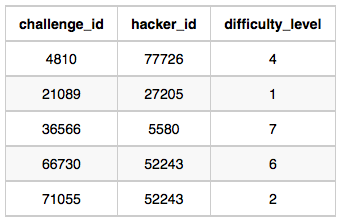
*Hackers* Table:



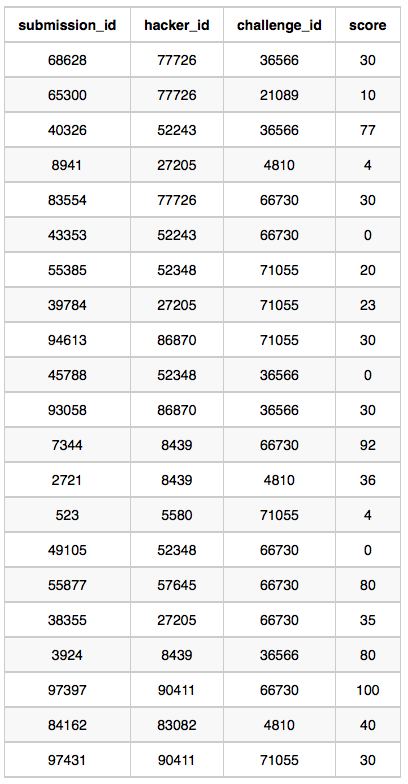
*Difficulty* Table:



*Challenges* Table:



*Submissions* Table:



Sample Output

90411 Joe

Explanation

Hacker *86870* got a score of *30* for challenge *71055* with a difficulty level of *2*, so *86870* earned a full score for this challenge.

Hacker *90411* got a score of *30* for challenge *71055* with a difficulty level of *2*, so *90411* earned a full score for this challenge.

Hacker *90411* got a score of *100* for challenge *66730* with a difficulty level of *6*, so *90411* earned a full score for this challenge.

Only hacker *90411* managed to earn a full score for more than one challenge, so we print the their *hacker\_id* and *name* as 2 space-separated values.

**Solution:**

SELECT h.hacker\_id, h.name

FROM Submissions AS s

JOIN Hackers AS h

ON s.hacker\_id = h.hacker\_id

JOIN Challenges AS c

ON s.challenge\_id = c.challenge\_id

JOIN Difficulty AS d

ON c.difficulty\_level = d.difficulty\_level

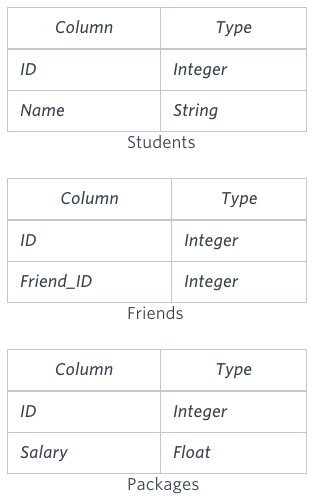
WHERE s.score = d.score

GROUP BY h.hacker\_id, h.name

HAVING COUNT(\*)>1

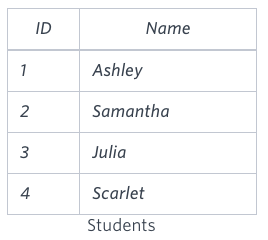
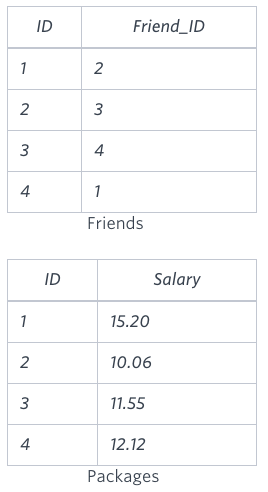
ORDER BY COUNT(\*) DESC, h.hacker\_id;

**Q36:** You are given three tables: *Students*,*Friends*and*Packages.* *Students* contains two columns: *ID* and *Name*. *Friends* contains two columns: *ID* and *Friend\_ID* (*ID* of the ONLY best friend). *Packages* contains two columns: *ID* and *Salary* (offered salary in $ thousands per month).

****

Write a query to output the names of those students whose best friends got offered a higher salary than them. Names must be ordered by the salary amount offered to the best friends. It is guaranteed that no two students got same salary offer.

Sample Input

** **

Sample Output

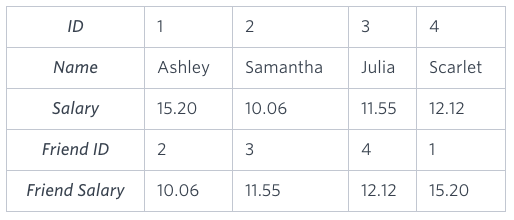
Samantha

Julia

Scarlet

Explanation

See the following table:

****

Now,

* *Samantha's* best friend got offered a higher salary than her at 11.55
* *Julia's* best friend got offered a higher salary than her at 12.12
* *Scarlet's* best friend got offered a higher salary than her at 15.2
* *Ashley's* best friend did NOT get offered a higher salary than her

The name output, when ordered by the salary offered to their friends, will be:

* *Samantha*
* *Julia*
* *Scarlet*

**Solution:**

SELECT s.Name FROM Students AS s

JOIN Packages AS sp ON s.ID = sp.ID

JOIN Friends AS f ON s.ID = f.ID

JOIN Packages AS fp ON f.Friend\_ID = fp.ID

WHERE sp.Salary < fp.Salary

ORDER BY fp.Salary;