

Amber's conglomerate corporation just acquired some new companies. Each of the companies follows this hierarchy:

Given the table schemas below, write a query to print the company\_code, founder name, total number of lead managers, total number of senior managers, total number of managers, and total number of employees. Order your output by ascending company\_code.

Note:

The tables may contain duplicate records.

The company\_code is string, so the sorting should not be numeric. For example, if the company\_codes are C\_1, C\_2, and C\_10, then the ascending company\_codes will be C\_1, C\_10, and C\_2.

Input Format

The following tables contain company data:

Company: The company\_code is the code of the company and founder is the founder of the company.

Lead\_Manager: The lead\_manager\_code is the code of the lead manager, and the company\_code is the code of the working company.

Senior\_Manager: The senior\_manager\_code is the code of the senior manager, the lead\_manager\_code is the code of its lead manager, and the company\_code is the code of the working company.

Manager: The manager\_code is the code of the manager, the senior\_manager\_code is the code of its senior manager, the lead\_manager\_code is the code of its lead manager, and the company\_code is the code of the working company.

Employee: The employee\_code is the code of the employee, the manager\_code is the code of its manager, the senior\_manager\_code is the code of its senior manager, the lead\_manager\_code is the code of its lead manager, and the company\_code is the code of the working company.

Sample Input

Company Table: Lead\_Manager Table: Senior\_Manager Table: Manager Table: Employee Table:

Sample Output

C1 Monika 1 2 1 2

C2 Samantha 1 1 2 2

Explanation

In company C1, the only lead manager is LM1. There are two senior managers, SM1 and SM2, under LM1. There is one manager, M1, under senior manager SM1. There are two employees, E1 and E2, under manager M1.

In company C2, the only lead manager is LM2. There is one senior manager, SM3, under LM2. There are two managers, M2 and M3, under senior manager SM3. There is one employee, E3, under manager M2, and another employee, E4, under manager, M3.

**Solution:**

SELECT c.company\_code,

c.founder,

COUNT(DISTINCT l.lead\_manager\_code),

COUNT(DISTINCT s.senior\_manager\_code),

COUNT(DISTINCT m.manager\_code),

COUNT(DISTINCT e.employee\_code)

FROM Company c

JOIN Lead\_Manager l ON c.company\_code = l.company\_code

JOIN Senior\_Manager s ON l.lead\_manager\_code = s.lead\_manager\_code

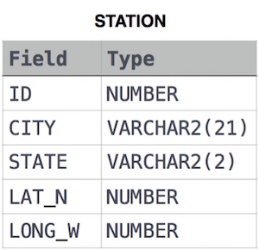
JOIN Manager m ON s.senior\_manager\_code = m.senior\_manager\_code

JOIN Employee e ON m.manager\_code = e.manager\_code

GROUP BY c.company\_code, c.founder

ORDER BY c.company\_code ASC

**Weather Observation Station 20**

A median is defined as a number separating the higher half of a data set from the lower half. Query the median of the Northern Latitudes (LAT\_N) from STATION and round your answer to decimal places.

Input Format

The STATION table is described as follows:

**Solution:**

**SELECT CAST(ROUND(LAT\_N,4) AS DECIMAL(10,4))**

**FROM STATION**

**WHERE LAT\_N = (SELECT DISTINCT PERCENTILE\_CONT(0.5)**

**WITHIN GROUP (ORDER BY LAT\_N) OVER() AS "Median"**

**FROM STATION)**

**The Report**

A screenshot of a number grid

Description automatically generatedYou are given two tables: Students and Grades. Students contains three columns ID, Name and Marks.

Grades contains the following data:A screenshot of a computer

Description automatically generated

Ketty gives Eve a task to generate a report containing three columns: Name, Grade and Mark. Ketty doesn't want the NAMES of those students who received a grade lower than 8. The report must be in descending order by grade -- i.e. higher grades are entered first. If there is more than one student with the same grade (8-10) assigned to them, order those particular students by their name alphabetically. Finally, if the grade is lower than 8, use "NULL" as their name and list them by their grades in descending order. If there is more than one student with the same grade (1-7) assigned to them, order those particular students by their marks in ascending order.

Write a query to help Eve.

**Solution:**

**WITH student\_grade AS (**

**SELECT S.Name AS name, G.Grade AS grade, S.Marks AS marks**

**FROM Students S, Grades G**

**WHERE marks BETWEEN G.Min\_Mark AND G.Max\_Mark**

**)**

**SELECT**

**CASE**

**WHEN grade < 8 THEN NULL**

**ELSE name**

**END,**

**grade,**

**marks**

**FROM student\_grade**

**ORDER BY grade DESC,name;**

**Top Competitors:**

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.

**Solution:**

**SELECT h.hacker\_id, h.name**

**FROM Hackers h**

**JOIN (**

**SELECT s.hacker\_id, COUNT(DISTINCT s.challenge\_id) AS num\_full\_scores**

**FROM Submissions s**

**JOIN Challenges c ON s.challenge\_id = c.challenge\_id**

**WHERE s.score = (SELECT MAX(score) FROM Difficulty WHERE difficulty\_level = c.difficulty\_level)**

**GROUP BY s.hacker\_id**

**HAVING COUNT(DISTINCT s.challenge\_id) > 1**

**) full\_scores ON h.hacker\_id = full\_scores.hacker\_id**

**ORDER BY full\_scores.num\_full\_scores DESC, h.hacker\_id;**

**Ollivander's Inventory:**

Harry Potter and his friends are at Ollivander's with Ron, finally replacing Charlie's old broken wand.

Hermione decides the best way to choose is by determining the minimum number of gold galleons needed to buy each non-evil wand of high power and age. Write a query to print the id, age, coins\_needed, and power of the wands that Ron's interested in, sorted in order of descending power. If more than one wand has same power, sort the result in order of descending age.

**Solution:**

**WITH valid\_wands AS (**

**SELECT w.id, wp.age, w.coins\_needed, w.power**

**FROM wands w INNER JOIN wands\_property wp ON w.code = wp.code**

**WHERE wp.is\_evil = 0**

**)**

**SELECT \***

**FROM valid\_wands vw**

**WHERE vw.coins\_needed <= (**

**SELECT MIN(w.coins\_needed)**

**FROM wands w**

**INNER JOIN wands\_property wp ON w.code = wp.code**

**WHERE wp.age = vw.age AND w.power = vw.power**

**)**

**ORDER BY vw.power DESC, vw.age DESC;**

**Challenges:**

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.

**SELECT c.hacker\_id, h.name, COUNT(c.challenge\_id)**

**FROM Hackers AS h**

**JOIN Challenges AS c ON h.hacker\_id = c.hacker\_id**

**GROUP BY c.hacker\_id, h.name**

**HAVING**

**COUNT(c.challenge\_id) = (SELECT TOP 1 COUNT(c1.challenge\_id)**

**FROM Challenges AS c1**

**GROUP BY c1.hacker\_id**

**ORDER BY COUNT(\*) DESC) OR**

**NOT EXISTS (**

**SELECT 1**

**FROM Challenges AS c2**

**WHERE c2.hacker\_id <> c.hacker\_id**

**GROUP BY c2.hacker\_id**

**HAVING COUNT(c2.challenge\_id) = COUNT(c.challenge\_id)**

**)**

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

**Contest Leaderboard:**

You did such a great job helping Julia with her last coding contest challenge that she wants you to work on this one, too!

The total score of a hacker is the sum of their maximum scores for all of the challenges. Write a query to print the hacker\_id, name, and total score of the hackers ordered by the descending score. If more than one hacker achieved the same total score, then sort the result by ascending hacker\_id. Exclude all hackers with a total score of 0 from your result.

**WITH valid\_hackers AS (**

**SELECT h.hacker\_id, h.name, s.challenge\_id, MAX(score) AS max\_score**

**FROM hackers h**

**INNER JOIN submissions s ON h.hacker\_id = s.hacker\_id**

**GROUP BY h.hacker\_id, h.name, s.challenge\_id**

**)**

**SELECT hacker\_id, name, SUM(max\_score) AS total\_score**

**FROM valid\_hackers**

**GROUP BY hacker\_id, name**

**HAVING SUM(max\_score) <> 0**

**ORDER BY total\_score DESC, hacker\_id ASC;**