|  |  |  |
| --- | --- | --- |
| **Question No.** | **Question Description** | **Solution** |
| New Companies | 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. | 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, Lead\_Manager l, Senior\_Manager s, Manager m, Employee e  where c.company\_code = l.company\_code  and l.lead\_manager\_code=s.lead\_manager\_code  and s.senior\_manager\_code=m.senior\_manager\_code  and m.manager\_code=e.manager\_code  group by c.company\_code, c.founder  order by c.company\_code, c.founder; |
| Symmetric Pairs | Print all symmetric pairs of (X1,Y1) and (X2,Y2) | SELECT f1.X, f1.Y  FROM Functions f1  JOIN Functions f2  ON f1.X=f2.Y AND f1.Y=f2.X  GROUP BY f1.X, f1.Y  HAVING COUNT(f1.X)>1 or f1.X<f1.Y  ORDER BY f1.X |
| Ollivander’s Inventory | Print Grade based on each individual’s marks | SELECT IF(GRADE < 8, NULL, NAME), GRADE, MARKS  FROM STUDENTS  JOIN GRADES  WHERE MARKS BETWEEN MIN\_MARK AND MAX\_MARK  ORDER BY GRADE DESC, NAME |
| Top Competitors | 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. | select hacker\_id, name from (  select h.hacker\_id, h.name, count(\*)  from hackers h  left join submissions s  on h.hacker\_id = s.hacker\_id  left join challenges c  on s.challenge\_id = c.challenge\_id  left join difficulty 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 asc  )a |
| 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 a.id, a.name, a.counter  FROM (  SELECT c.hacker\_id id, h.name name, count(c.hacker\_id) counter  FROM Hackers h  JOIN Challenges c on c.hacker\_id = h.hacker\_id  GROUP BY c.hacker\_id, h.name  )a  WHERE a.counter = (SELECT max(b.counter) FROM (  SELECT c.hacker\_id id, h.name name, count(c.hacker\_id) counter  FROM Hackers h  JOIN Challenges c on c.hacker\_id = h.hacker\_id  GROUP BY c.hacker\_id, h.name)b  )  OR  a.counter in (SELECT d.counter from (SELECT c.hacker\_id id, h.name name, count(c.hacker\_id)counter  FROM Hackers h  JOIN Challenges c on c.hacker\_id = h.hacker\_id  GROUP BY c.hacker\_id, h.name)d  GROUP BY d.counter  HAVING COUNT(d.counter)=1  )  ORDER BY a.counter desc, id; |
| Contest Leaderboard | 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 0 of from your result. | select hacker\_id, name, sum(score)  from (  select s.hacker\_id, h.name, challenge\_id, max(score) as score  from submissions s  left join hackers h  on s.hacker\_id = h.hacker\_id  group by s.hacker\_id, h.name, challenge\_id)a  group by hacker\_id, name  having sum(score) > 0  order by sum(score) desc, hacker\_id asc; |
| SQL Project Planning | Write a query to output the start and end dates of projects listed by the number of days it took to complete the project in ascending order. If there is more than one project that have the same number of completion days, then order by the start date of the project. | SELECT Start\_Date, MIN(End\_Date)  FROM  (SELECT Start\_Date  FROM Projects  WHERE Start\_Date NOT IN (SELECT End\_Date FROM Projects)) a,    (SELECT end\_date  FROM PROJECTS  WHERE end\_date NOT IN (SELECT start\_date FROM PROJECTS)) b  where start\_date < end\_date  GROUP BY start\_date  ORDER BY datediff(start\_date, MIN(end\_date)) DESC, start\_date |
| Placements | 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. | select s1.name  from students s1  join friends f1  on s1.id = f1.id  join packages p1  on s1.id = p1.id  join packages p2  on f1.friend\_id = p2.id  where p2.salary > p1.salary  order by p2.salary; |
|  |  |  |