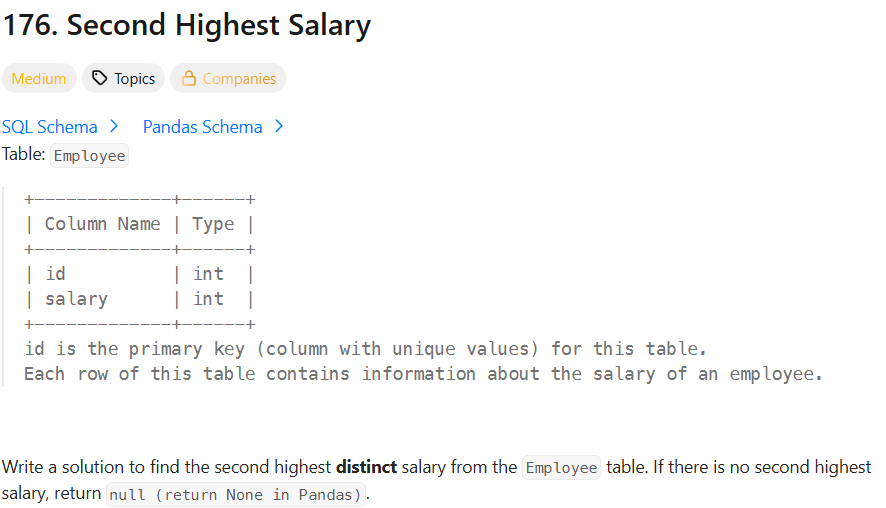


**Week -6**

**Q1.**

****

SOL.

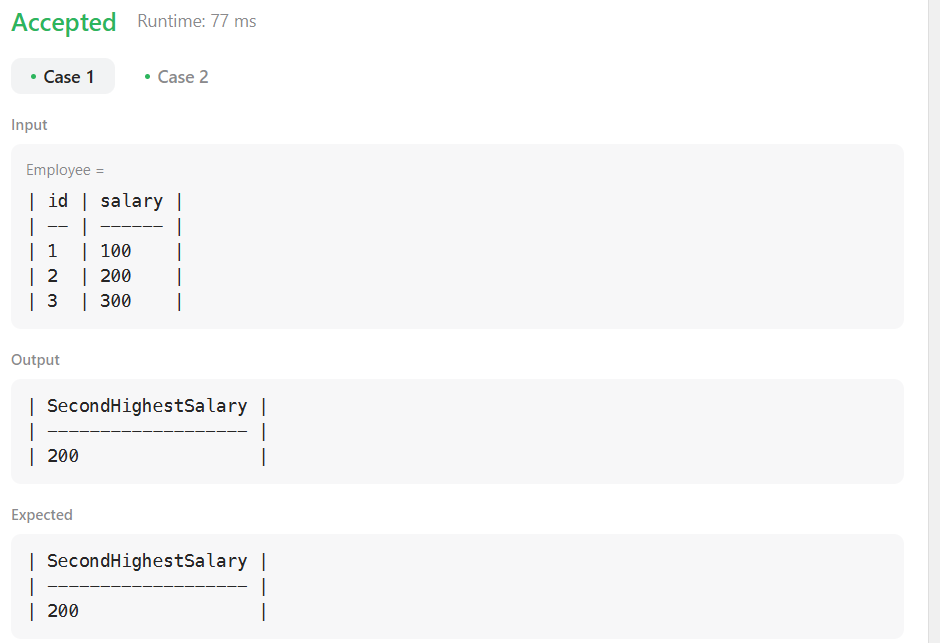
SELECT

(SELECT DISTINCT salary

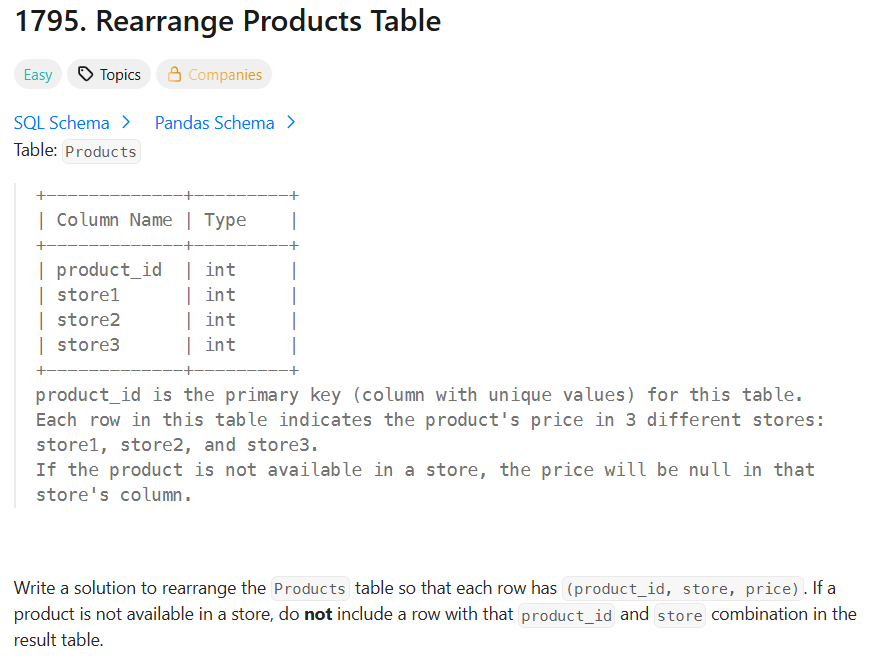
FROM Employee

ORDER BY salary DESC

LIMIT 1 OFFSET 1) AS SecondHighestSalary;



**Q2.**



SOL.

SELECT product\_id, 'store1' AS store, store1 AS price

FROM Products

WHERE store1 IS NOT NULL

UNION ALL

SELECT product\_id, 'store2' AS store, store2 AS price

FROM Products

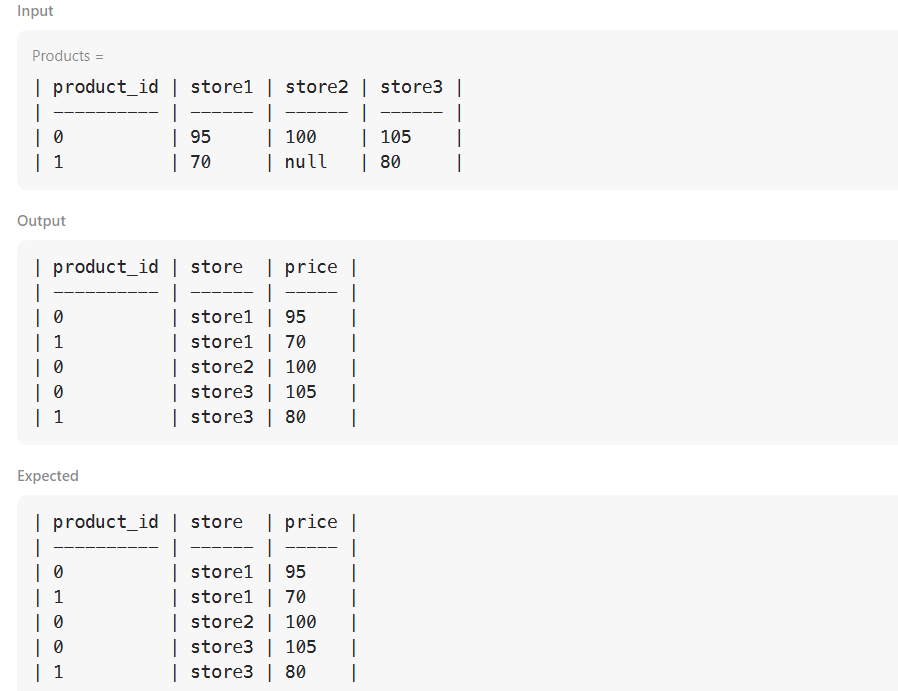
WHERE store2 IS NOT NULL

UNION ALL

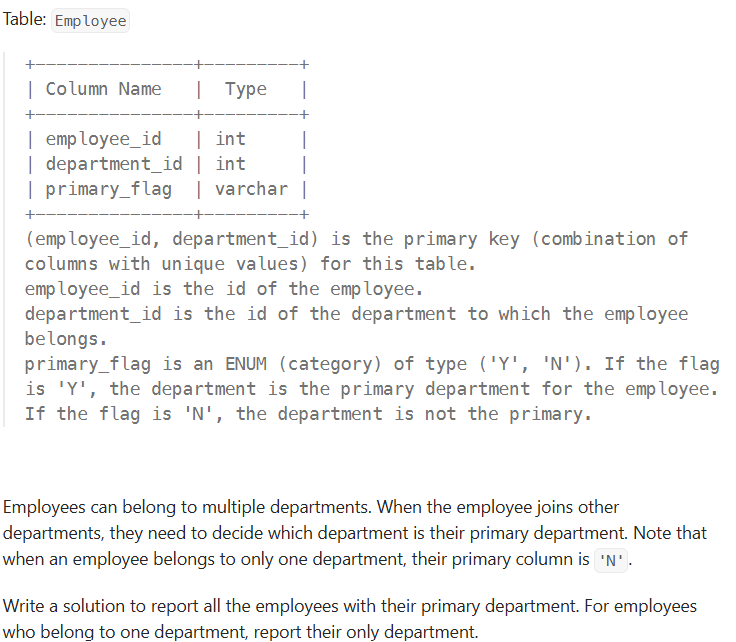
SELECT product\_id, 'store3' AS store, store3 AS price

FROM Products

WHERE store3 IS NOT NULL;



**Q3.**



SOL.

SELECT employee\_id, department\_id

FROM Employee

WHERE primary\_flag = 'Y'

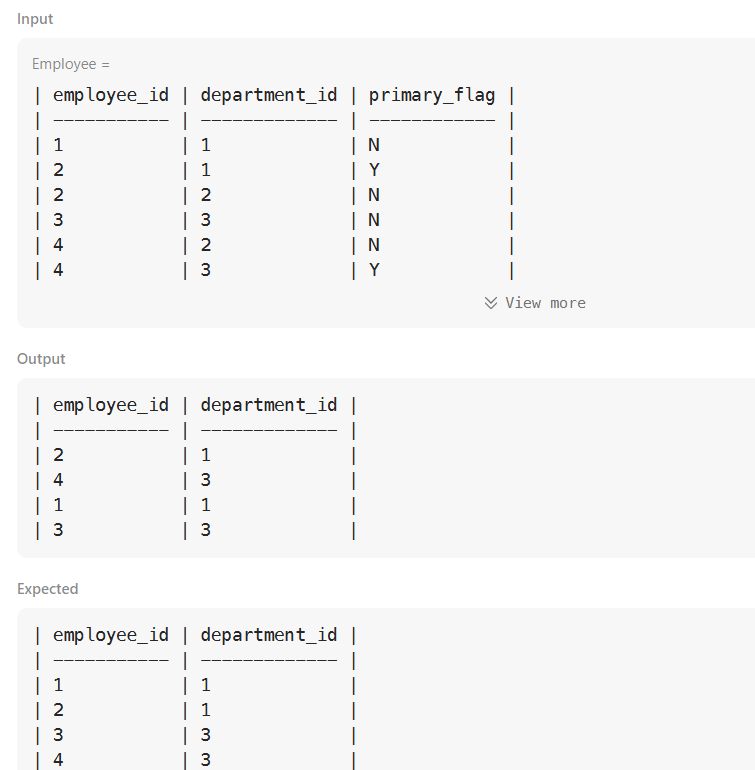
UNION

SELECT employee\_id, department\_id

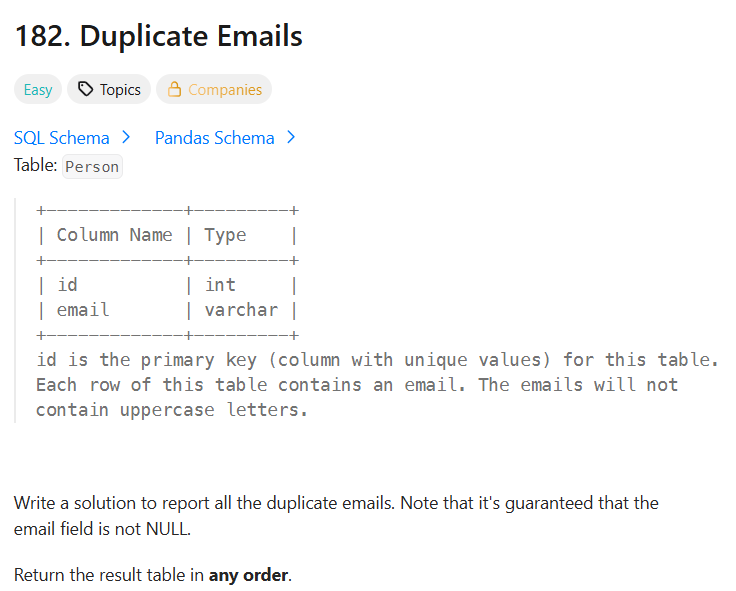
FROM Employee

GROUP BY employee\_id

HAVING COUNT(\*) = 1;



**Q4.**



SOL.

SELECT email AS Email

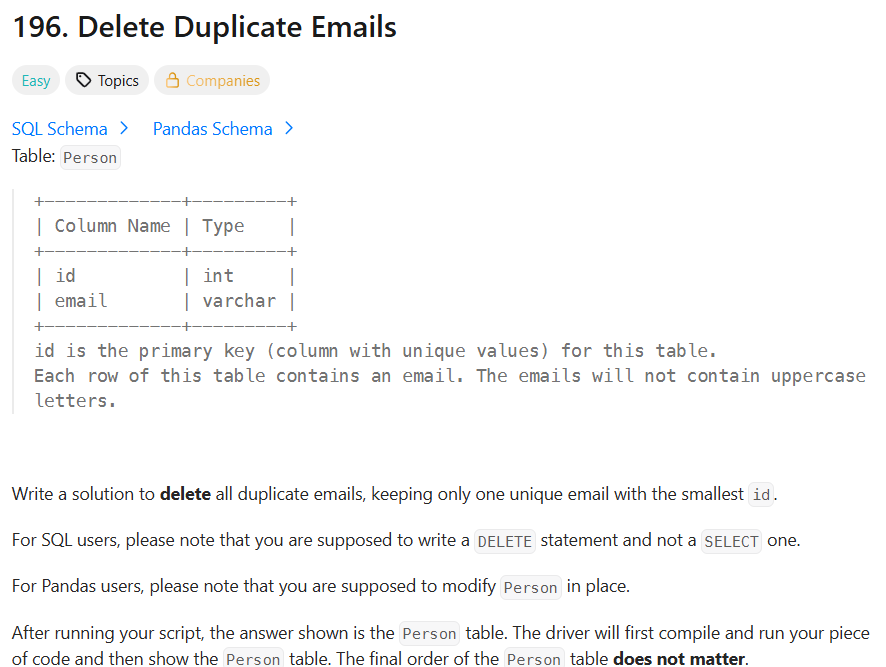
FROM Person

GROUP BY email

HAVING COUNT(\*) > 1;



**Q5.**



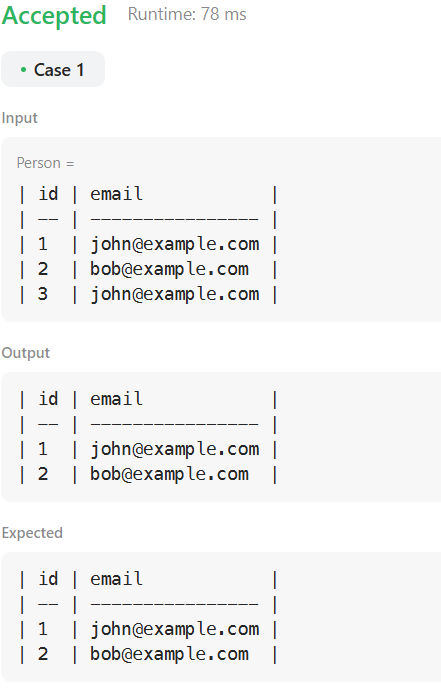
SOL.

DELETE p1

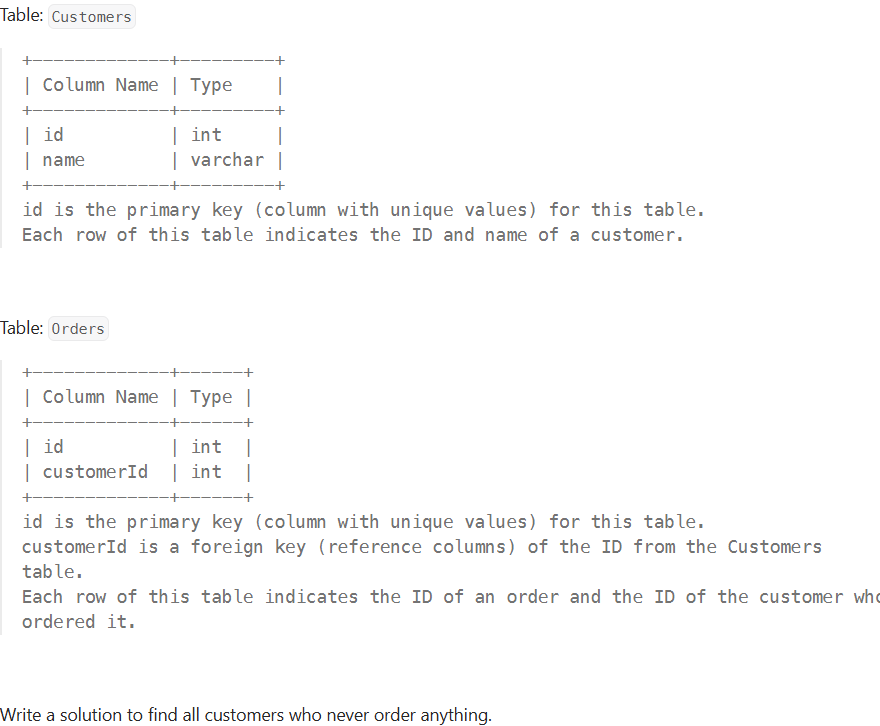
FROM Person p1

JOIN Person p2

ON p1.email = p2.email AND p1.id > p2.id;



**Q6**.



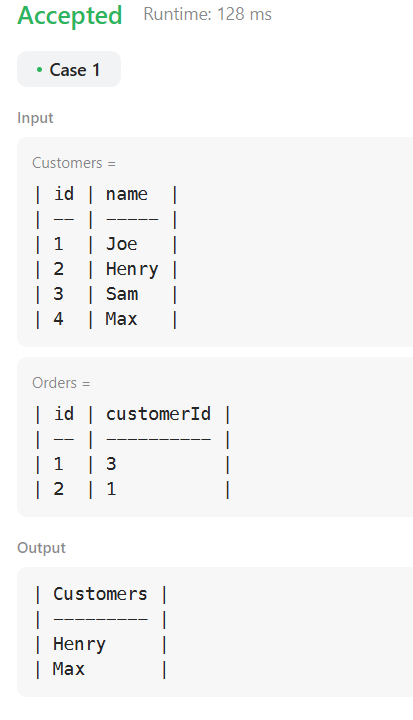
SOL.

SELECT c.name AS Customers

FROM Customers c

LEFT JOIN Orders o ON c.id = o.customerId

WHERE o.id IS NULL;



**Q7.**

A company's executives are interested in seeing who earns the most money in each of the company's departments. A **high earner** in a department is an employee who has a salary in the **top three unique** salaries for that department.

Write a solution to find the employees who are **high earners** in each of the departments.

Return the result table **in any order**.

SOL

# Write your MySQL query statement below

Select d.name as Department , e.name as Employee , e.salary as Salary

from

(

    SELECT \*,

           DENSE\_RANK() OVER (PARTITION BY departmentId ORDER BY salary DESC) AS salary\_rank

    FROM Employee

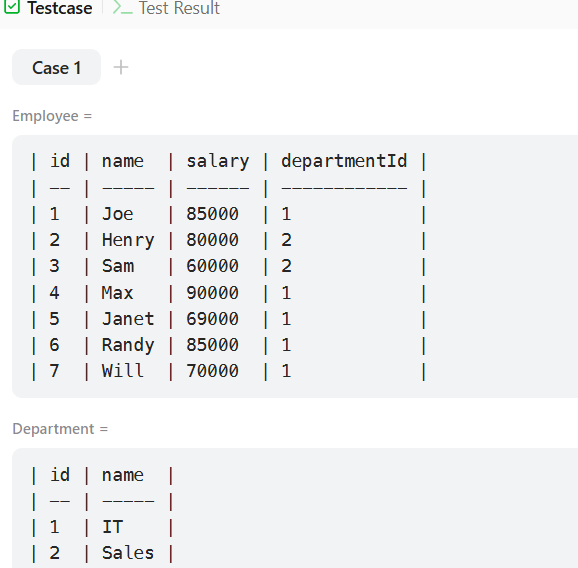
)

e

join Department d

on e.departmentId= d.id

where e.salary\_rank <= 3;



**Q8.**

Write a solution to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in **any order**.

SOL.

SELECT

    p.firstName,

    p.lastName,

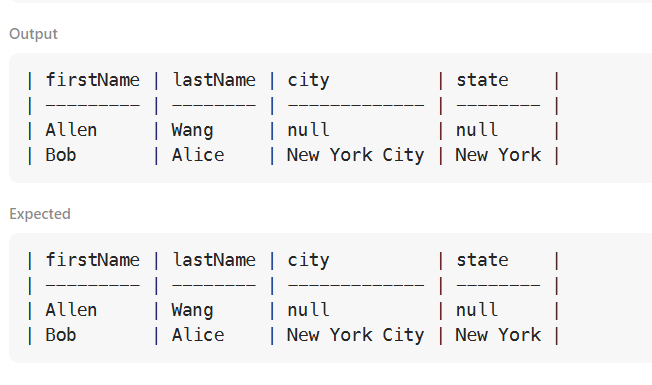
    a.city,

    a.state

FROM Person p

LEFT JOIN Address a

    ON p.personId = a.personId;



**Q9.**

Write a solution to find the employees who earn more than their managers.

Return the result table in **any order**.

SOL

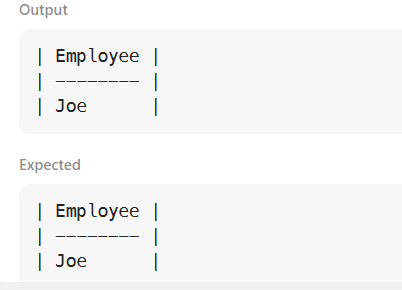
SELECT e1.name AS Employee

FROM Employee e1

JOIN Employee e2

  ON e1.managerId = e2.id

WHERE e1.salary > e2.salary;



**Q10.**

Write a solution to report all the duplicate emails. Note that it's guaranteed that the email field is not NULL.

Return the result table in **any order**.

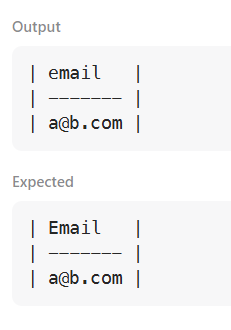
SOL

SELECT email

FROM Person

GROUP BY email

HAVING COUNT(\*) > 1;



**Q11.**

Find all numbers that appear at least three times consecutively.

Return the result table in **any order**.

SOL

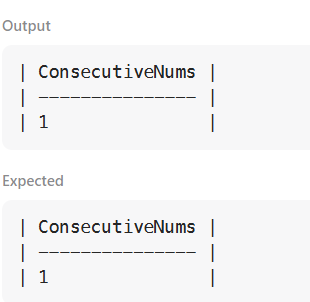
SELECT DISTINCT l1.Num AS ConsecutiveNums

FROM Logs l1

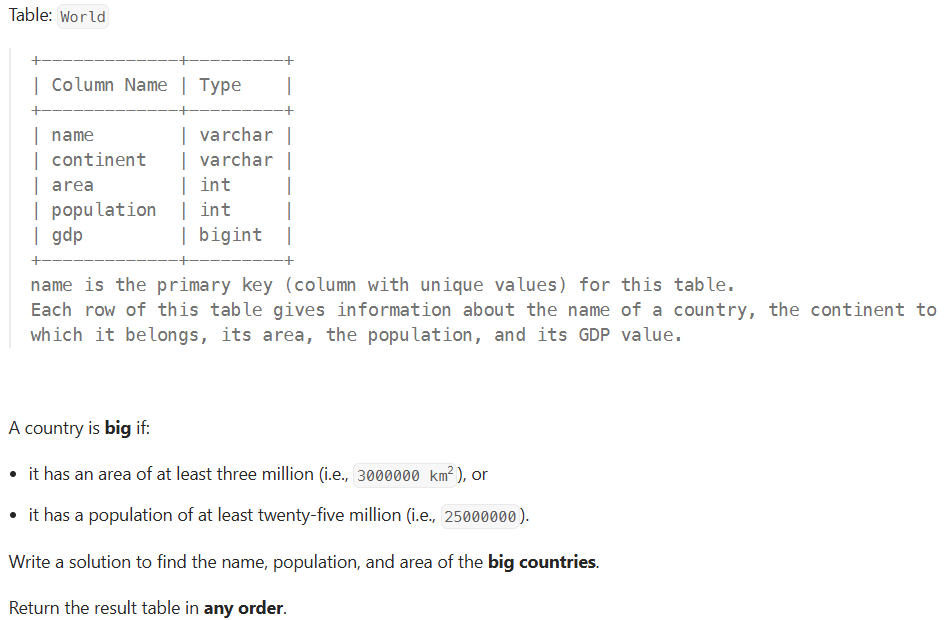
JOIN Logs l2 ON l1.Id = l2.Id - 1

JOIN Logs l3 ON l1.Id = l3.Id - 2

WHERE l1.Num = l2.Num AND l1.Num = l3.Num;



**Q12.**

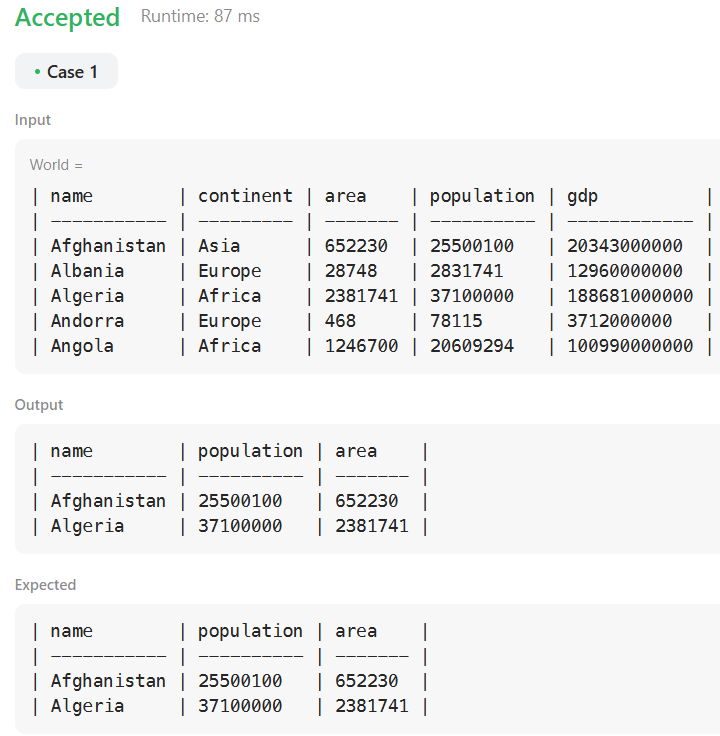


SOL.

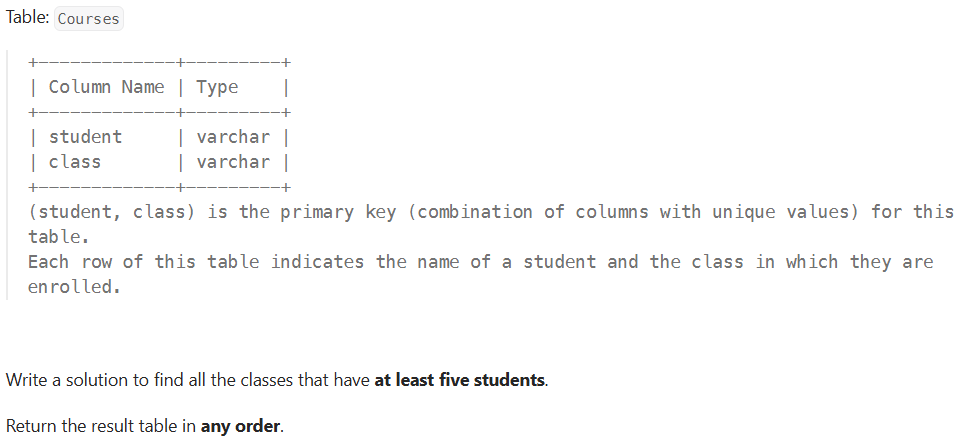
SELECT name, population, area

FROM World

WHERE area >= 3000000 OR population >= 25000000;



**Q13.**

****

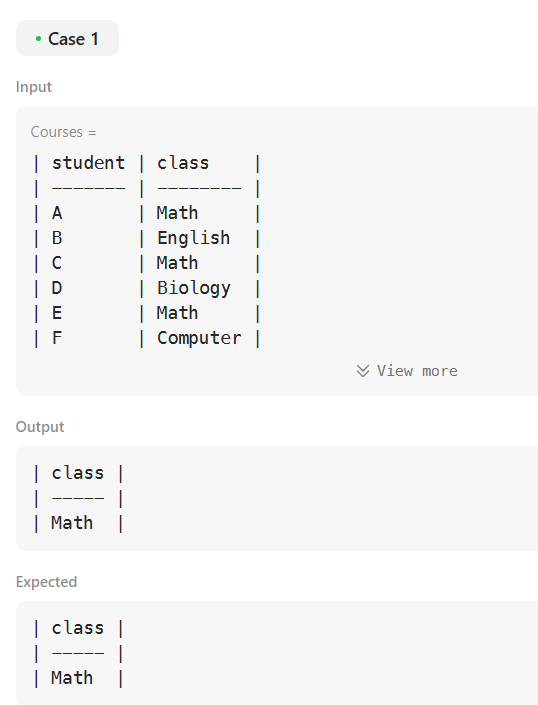
**SOL.**

SELECT class

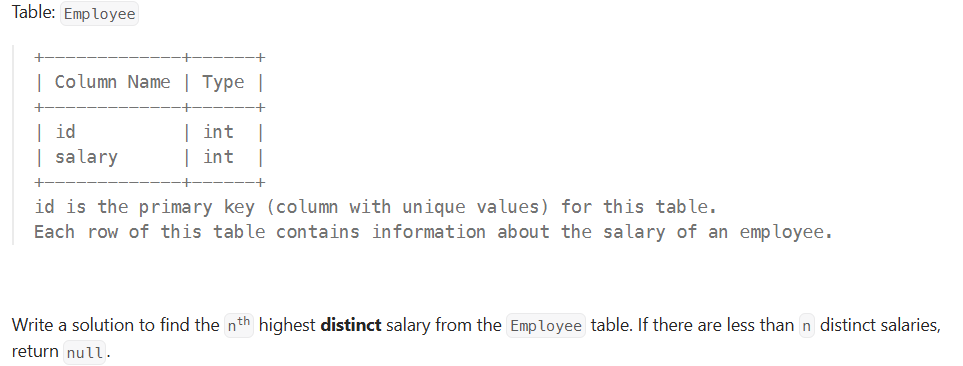
FROM Courses

GROUP BY class

HAVING COUNT(student) >= 5;



Q14.



SOL.

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

RETURN (

SELECT DISTINCT salary

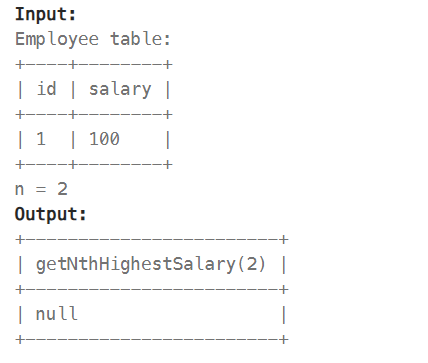
FROM Employee

ORDER BY salary DESC

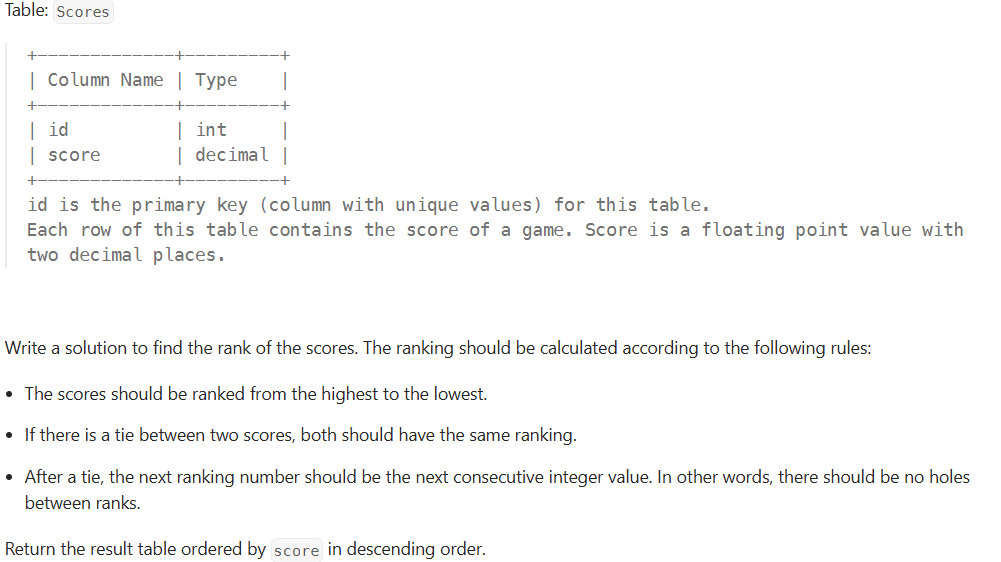
LIMIT 1 OFFSET N - 1

);

END



Q15.



SOL.

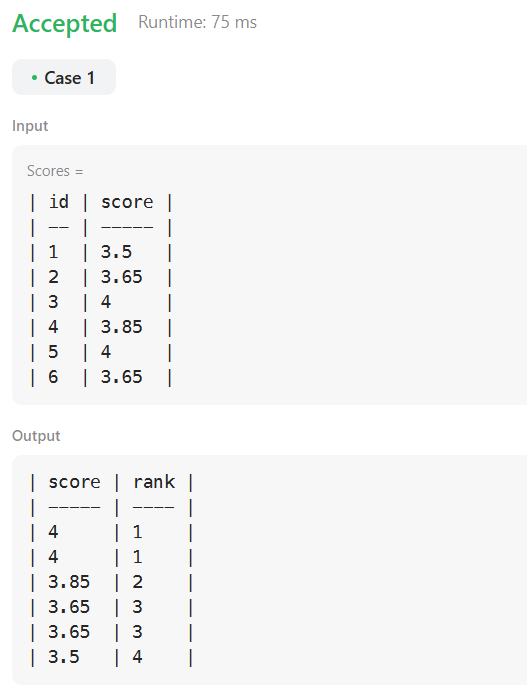
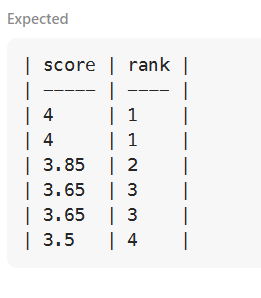
SELECT

score,

DENSE\_RANK() OVER (ORDER BY score DESC) AS `rank`

FROM Scores

ORDER BY score DESC;

-SUCHARITA GORAI