1)Write a solution to swap the seat id of every two consecutive students. If the number of students is odd, the id of the last student is not swapped.

Return the result table ordered by id in ascending order.

Example 1:

Input:

Seat table:

+----+---------+

| id | student |

+----+---------+

| 1 | Abbot |

| 2 | Doris |

| 3 | Emerson |

| 4 | Green |

| 5 | Jeames |

+----+---------+

Output:

+----+---------+

| id | student |

+----+---------+

| 1 | Doris |

| 2 | Abbot |

| 3 | Green |

| 4 | Emerson |

| 5 | Jeames |

+----+---------+

ANS :

SELECT

CASE

WHEN id = (SELECT MAX(id) FROM seat) AND id % 2 = 1

THEN id

WHEN id % 2 = 1

THEN id + 1

ELSE id - 1

END AS id,

student

FROM seat

ORDER BY id

2)You are the restaurant owner and you want to analyze a possible expansion (there will be at least one customer every day).

Compute the moving average of how much the customer paid in a seven days window (i.e., current day + 6 days before). average\_amount should be rounded to two decimal places.

Return the result table ordered by visited\_on in ascending order.

Example 1:

Input:

Customer table:

+-------------+--------------+--------------+-------------+

| customer\_id | name | visited\_on | amount |

+-------------+--------------+--------------+-------------+

| 1 | Jhon | 2019-01-01 | 100 |

| 2 | Daniel | 2019-01-02 | 110 |

| 3 | Jade | 2019-01-03 | 120 |

| 4 | Khaled | 2019-01-04 | 130 |

| 5 | Winston | 2019-01-05 | 110 |

| 6 | Elvis | 2019-01-06 | 140 |

| 7 | Anna | 2019-01-07 | 150 |

| 8 | Maria | 2019-01-08 | 80 |

| 9 | Jaze | 2019-01-09 | 110 |

| 1 | Jhon | 2019-01-10 | 130 |

| 3 | Jade | 2019-01-10 | 150 |

+-------------+--------------+--------------+-------------+

Output:

+--------------+--------------+----------------+

| visited\_on | amount | average\_amount |

+--------------+--------------+----------------+

| 2019-01-07 | 860 | 122.86 |

| 2019-01-08 | 840 | 120 |

| 2019-01-09 | 840 | 120 |

| 2019-01-10 | 1000 | 142.86 |

+--------------+--------------+----------------+

ANS :

SELECT

visited\_on,

(

SELECT SUM(amount)

FROM customer

WHERE visited\_on BETWEEN DATE\_SUB(c.visited\_on, INTERVAL 6 DAY) AND c.visited\_on

) AS amount,

ROUND(

(

SELECT SUM(amount) / 7

FROM customer

WHERE visited\_on BETWEEN DATE\_SUB(c.visited\_on, INTERVAL 6 DAY) AND c.visited\_on

),

2

) AS average\_amount

FROM customer c

WHERE visited\_on >= (

SELECT DATE\_ADD(MIN(visited\_on), INTERVAL 6 DAY)

FROM customer

)

GROUP BY visited\_on;

3)Write a solution to fix the names so that only the first character is uppercase and the rest are lowercase.

Return the result table ordered by user\_id.

Example 1:

Input:

Users table:

+---------+-------+

| user\_id | name |

+---------+-------+

| 1 | aLice |

| 2 | bOB |

+---------+-------+

Output:

+---------+-------+

| user\_id | name |

+---------+-------+

| 1 | Alice |

| 2 | Bob |

+---------+-------+

ANS :

SELECT user\_id , CONCAT(UPPER(SUBSTR(name,1,1)),LOWER(SUBSTR(name,2))) AS name

FROM Users

ORDER BY

Users.user\_id ASC

4)Write a solution to delete all duplicate emails, keeping only one unique email with the smallest id.

For SQL users, please note that you are supposed to write a DELETE statement and not a SELECT one.

Example 1:

Input:

Person table:

+----+------------------+

| id | email |

+----+------------------+

| 1 | john@example.com |

| 2 | bob@example.com |

| 3 | john@example.com |

+----+------------------+

Output:

+----+------------------+

| id | email |

+----+------------------+

| 1 | john@example.com |

| 2 | bob@example.com |

+----+------------------+

ANS :

DELETE p1

FROM Person p1

INNER JOIN Person p2

WHERE p1.email = p2.email AND

p1.id > p2.id;

5)Write a solution to find the second highest salary from the Employee table. If there is no second highest salary, return null

The result format is in the following example.

Example 1:

Input:

Employee table:

+----+--------+

| id | salary |

+----+--------+

| 1 | 100 |

| 2 | 200 |

| 3 | 300 |

+----+--------+

Output:

+---------------------+

| SecondHighestSalary |

+---------------------+

| 200 |

+---------------------+

Example 2:

Input:

Employee table:

+----+--------+

| id | salary |

+----+--------+

| 1 | 100 |

+----+--------+

Output:

+---------------------+

| SecondHighestSalary |

+---------------------+

| null |

+---------------------+

ANS :

# Write your MySQL query statement below

-- SELECT MAX(e1.salary) AS SecondHighestSalary

-- FROM Employee e1

-- INNER JOIN Employee e2

-- ON e1.salary < e2.salary;

SELECT MAX(salary) AS SecondHighestSalary

FROM employee

WHERE salary <

(SELECT MAX(salary) FROM employee);

6)Write a solution to find the patient\_id, patient\_name, and conditions of the patients who have Type I Diabetes. Type I Diabetes always starts with DIAB1 prefix.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Patients table:

+------------+--------------+--------------+

| patient\_id | patient\_name | conditions |

+------------+--------------+--------------+

| 1 | Daniel | YFEV COUGH |

| 2 | Alice | |

| 3 | Bob | DIAB100 MYOP |

| 4 | George | ACNE DIAB100 |

| 5 | Alain | DIAB201 |

+------------+--------------+--------------+

Output:

+------------+--------------+--------------+

| patient\_id | patient\_name | conditions |

+------------+--------------+--------------+

| 3 | Bob | DIAB100 MYOP |

| 4 | George | ACNE DIAB100 |

+------------+--------------+--------------+

ANS :

SELECT patient\_id, patient\_name, conditions

FROM Patients

WHERE conditions LIKE 'DIAB1%' OR conditions LIKE '% DIAB1%';

7)Write a solution to find for each date the number of different products sold and their names.

The sold products names for each date should be sorted lexicographically.

Return the result table ordered by sell\_date.

The result format is in the following example.

Example 1:

Input:

Activities table:

+------------+------------+

| sell\_date | product |

+------------+------------+

| 2020-05-30 | Headphone |

| 2020-06-01 | Pencil |

| 2020-06-02 | Mask |

| 2020-05-30 | Basketball |

| 2020-06-01 | Bible |

| 2020-06-02 | Mask |

| 2020-05-30 | T-Shirt |

+------------+------------+

Output:

+------------+----------+------------------------------+

| sell\_date | num\_sold | products |

+------------+----------+------------------------------+

| 2020-05-30 | 3 | Basketball,Headphone,T-shirt |

| 2020-06-01 | 2 | Bible,Pencil |

| 2020-06-02 | 1 | Mask |

+------------+----------+---------------------------

ANS :

SELECT

sell\_date,

COUNT(DISTINCT product) AS num\_sold,

GROUP\_CONCAT(DISTINCT product ORDER BY product SEPARATOR ',') AS products

FROM activities

GROUP BY sell\_date

ORDER BY sell\_date ASC;

8)Write a solution to get the names of products that have at least 100 units ordered in February 2020 and their amount.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Products table:

+-------------+-----------------------+------------------+

| product\_id | product\_name | product\_category |

+-------------+-----------------------+------------------+

| 1 | Leetcode Solutions | Book |

| 2 | Jewels of Stringology | Book |

| 3 | HP | Laptop |

| 4 | Lenovo | Laptop |

| 5 | Leetcode Kit | T-shirt |

+-------------+-----------------------+------------------+

Orders table:

+--------------+--------------+----------+

| product\_id | order\_date | unit |

+--------------+--------------+----------+

| 1 | 2020-02-05 | 60 |

| 1 | 2020-02-10 | 70 |

| 2 | 2020-01-18 | 30 |

| 2 | 2020-02-11 | 80 |

| 3 | 2020-02-17 | 2 |

| 3 | 2020-02-24 | 3 |

| 4 | 2020-03-01 | 20 |

| 4 | 2020-03-04 | 30 |

| 4 | 2020-03-04 | 60 |

| 5 | 2020-02-25 | 50 |

| 5 | 2020-02-27 | 50 |

| 5 | 2020-03-01 | 50 |

+--------------+--------------+----------+

Output:

+--------------------+---------+

| product\_name | unit |

+--------------------+---------+

| Leetcode Solutions | 130 |

| Leetcode Kit | 100 |

+--------------------+---------+

ANS :

SELECT product\_name, SUM(unit) AS unit

FROM Products

INNER JOIN Orders

USING(product\_id)

WHERE MONTH(order\_date) = 2 AND YEAR(order\_date) = 2020

GROUP BY product\_id

HAVING unit >= 100;

9)Write a solution to find the users who have valid emails.

A valid e-mail has a prefix name and a domain where:

The prefix name is a string that may contain letters (upper or lower case), digits, underscore '\_', period '.', and/or dash '-'. The prefix name must start with a letter.

The domain is '@leetcode.com'.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Users table:

+---------+-----------+-------------------------+

| user\_id | name | mail |

+---------+-----------+-------------------------+

| 1 | Winston | winston@leetcode.com |

| 2 | Jonathan | jonathanisgreat |

| 3 | Annabelle | bella-@leetcode.com |

| 4 | Sally | sally.come@leetcode.com |

| 5 | Marwan | quarz#2020@leetcode.com |

| 6 | David | david69@gmail.com |

| 7 | Shapiro | .shapo@leetcode.com |

+---------+-----------+-------------------------+

Output:

+---------+-----------+-------------------------+

| user\_id | name | mail |

+---------+-----------+-------------------------+

| 1 | Winston | winston@leetcode.com |

| 3 | Annabelle | bella-@leetcode.com |

| 4 | Sally | sally.come@leetcode.com |

+---------+-----------+-------------------------+

ANS :

SELECT \*

FROM Users

WHERE mail REGEXP '^[A-Za-z][A-Za-z0-9\_\.\-]\*@leetcode(\\?com)?\\.com$';

10)Write a solution to report the ids and the names of all managers, the number of employees who report directly to them, and the average age of the reports rounded to the nearest integer.

Return the result table ordered by employee\_id.

The result format is in the following example.

Example 1:

Input:

Employees table:

+-------------+---------+------------+-----+

| employee\_id | name | reports\_to | age |

+-------------+---------+------------+-----+

| 9 | Hercy | null | 43 |

| 6 | Alice | 9 | 41 |

| 4 | Bob | 9 | 36 |

| 2 | Winston | null | 37 |

+-------------+---------+------------+-----+

Output:

+-------------+-------+---------------+-------------+

| employee\_id | name | reports\_count | average\_age |

+-------------+-------+---------------+-------------+

| 9 | Hercy | 2 | 39 |

+-------------+-------+---------------+-------------+

ANS:

SELECT

emp1.employee\_id,

emp1.name,

COUNT(emp2.employee\_id) AS reports\_count,

ROUND(AVG(emp2.age)) AS average\_age

FROM Employees emp1

INNER JOIN Employees emp2 ON emp1.employee\_id = emp2.reports\_to

GROUP BY emp1.employee\_id

ORDER BY emp1.employee\_id;

11)Write a solution to report all the employees with their primary department. For employees who belong to one department, report their only department.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Employee table:

+-------------+---------------+--------------+

| employee\_id | department\_id | primary\_flag |

+-------------+---------------+--------------+

| 1 | 1 | N |

| 2 | 1 | Y |

| 2 | 2 | N |

| 3 | 3 | N |

| 4 | 2 | N |

| 4 | 3 | Y |

| 4 | 4 | N |

+-------------+---------------+--------------+

Output:

+-------------+---------------+

| employee\_id | department\_id |

+-------------+---------------+

| 1 | 1 |

| 2 | 1 |

| 3 | 3 |

| 4 | 3 |

+-------------+---------------+

ANS :

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(employee\_id) = 1

ORDER BY employee\_id;

12)Write a solution to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.

Return the result table sorted in any order.

The result format is in the following example.

Example 1:

Input:

Visits

+----------+-------------+

| visit\_id | customer\_id |

+----------+-------------+

| 1 | 23 |

| 2 | 9 |

| 4 | 30 |

| 5 | 54 |

| 6 | 96 |

| 7 | 54 |

| 8 | 54 |

+----------+-------------+

Transactions

+----------------+----------+--------+

| transaction\_id | visit\_id | amount |

+----------------+----------+--------+

| 2 | 5 | 310 |

| 3 | 5 | 300 |

| 9 | 5 | 200 |

| 12 | 1 | 910 |

| 13 | 2 | 970 |

+----------------+----------+--------+

Output:

+-------------+----------------+

| customer\_id | count\_no\_trans |

+-------------+----------------+

| 54 | 2 |

| 30 | 1 |

| 96 | 1 |

+-------------+----------------+

ANS :

SELECT v.customer\_id, COUNT(v.visit\_id) AS count\_no\_trans

from Visits v

LEFT JOIN Transactions t

ON v.visit\_id = t.visit\_id

WHERE t.transaction\_id IS NULL

GROUP BY v.customer\_id;

13)Write a solution to report all the employees with their primary department. For employees who belong to one department, report their only department.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Employee table:

+-------------+---------------+--------------+

| employee\_id | department\_id | primary\_flag |

+-------------+---------------+--------------+

| 1 | 1 | N |

| 2 | 1 | Y |

| 2 | 2 | N |

| 3 | 3 | N |

| 4 | 2 | N |

| 4 | 3 | Y |

| 4 | 4 | N |

+-------------+---------------+--------------+

Output:

+-------------+---------------+

| employee\_id | department\_id |

+-------------+---------------+

| 1 | 1 |

| 2 | 1 |

| 3 | 3 |

| 4 | 3 |

+-------------+---------------+

ANS :

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(employee\_id) = 1

ORDER BY employee\_id;

14)Write a solution to find the confirmation rate of each user.

Return the result table in any order.

The result format is in the following example.

Example 1:

Input:

Signups table:

+---------+---------------------+

| user\_id | time\_stamp |

+---------+---------------------+

| 3 | 2020-03-21 10:16:13 |

| 7 | 2020-01-04 13:57:59 |

| 2 | 2020-07-29 23:09:44 |

| 6 | 2020-12-09 10:39:37 |

+---------+---------------------+

Confirmations table:

+---------+---------------------+-----------+

| user\_id | time\_stamp | action |

+---------+---------------------+-----------+

| 3 | 2021-01-06 03:30:46 | timeout |

| 3 | 2021-07-14 14:00:00 | timeout |

| 7 | 2021-06-12 11:57:29 | confirmed |

| 7 | 2021-06-13 12:58:28 | confirmed |

| 7 | 2021-06-14 13:59:27 | confirmed |

| 2 | 2021-01-22 00:00:00 | confirmed |

| 2 | 2021-02-28 23:59:59 | timeout |

+---------+---------------------+-----------+

Output:

+---------+-------------------+

| user\_id | confirmation\_rate |

+---------+-------------------+

| 6 | 0.00 |

| 3 | 0.00 |

| 7 | 1.00 |

| 2 | 0.50 |

+---------+-------------------+

ANS:

select s.user\_id, round(avg(if(c.action="confirmed",1,0)),2) as confirmation\_rate

from Signups as s left join Confirmations as c on s.user\_id= c.user\_id group by user\_id;

15)

Write a solution to report the fraction of players that logged in again on the day after the day they first logged in, rounded to 2 decimal places. In other words, you need to count the number of players that logged in for at least two consecutive days starting from their first login date, then divide that number by the total number of players.

The result format is in the following example.

Example 1:

Input:

Activity table:

+-----------+-----------+------------+--------------+

| player\_id | device\_id | event\_date | games\_played |

+-----------+-----------+------------+--------------+

| 1 | 2 | 2016-03-01 | 5 |

| 1 | 2 | 2016-03-02 | 6 |

| 2 | 3 | 2017-06-25 | 1 |

| 3 | 1 | 2016-03-02 | 0 |

| 3 | 4 | 2018-07-03 | 5 |

+-----------+-----------+------------+--------------+

Output:

+-----------+

| fraction |

+-----------+

| 0.33 |

+-----------+

ANS :

'SELECT

ROUND(COUNT(DISTINCT player\_id) / (SELECT COUNT(DISTINCT player\_id) FROM Activity), 2) AS fraction

FROM

Activity

WHERE

(player\_id, DATE\_SUB(event\_date, INTERVAL 1 DAY))

IN (

SELECT player\_id, MIN(event\_date) AS first\_login FROM Activity GROUP BY player\_id

)'