*/\*   
608. Tree Node   
Given a table tree, id is identifier of the tree node and p\_id is its parent node's id.   
  
+----+------+   
| id | p\_id |   
+----+------+   
| 1  | null |   
| 2  | 1    |   
| 3  | 1    |   
| 4  | 2    |   
| 5  | 2    |   
+----+------+   
Each node in the tree can be one of three types:   
Leaf: if the node is a leaf node.   
Root: if the node is the root of the tree.   
Inner: If the node is neither a leaf node nor a root node.   
Write a query to print the node id and the type of the node. Sort your output by the node id. The result for the above sample is:   
+----+------+   
| id | Type |   
+----+------+   
| 1  | Root |   
| 2  | Inner|   
| 3  | Leaf |   
| 4  | Leaf |   
| 5  | Leaf |   
+----+------+   
Explanation   
  
Node '1' is root node, because its parent node is NULL and it has child node '2' and '3'.   
Node '2' is inner node, because it has parent node '1' and child node '4' and '5'.   
Node '3', '4' and '5' is Leaf node, because they have parent node and they don't have child node.   
  
And here is the image of the sample tree as below:   
        1   
      /   \   
                      2       3   
                    /   \   
                  4       5   
Note   
  
If there is only one node on the tree, you only need to output its root attributes.   
\*/*   
SELECT id,   
       CASE   
         WHEN p\_id IS NULL THEN 'Root'   
         WHEN childs = 0 THEN 'Leaf'   
         ELSE 'Inner'   
       END AS Type   
FROM   (SELECT id,   
               p\_id,   
               (SELECT *Count*(\*)   
                FROM   tree t2   
                WHERE  t1.id = t2.p\_id) childs   
        FROM   tree t1) t1   
ORDER  BY id   
  
*/\*   
610. Triangle Judgement   
A pupil Tim gets homework to identify whether three line segments could possibly form a triangle.   
  
However, this assignment is very heavy because there are hundreds of records to calculate.   
Could you help Tim by writing a query to judge whether these three sides can form a triangle, assuming table triangle holds the length of the three sides x, y and z.   
| x  | y  | z  |   
|----|----|----|   
| 13 | 15 | 30 |   
| 10 | 20 | 15 |   
For the sample data above, your query should return the follow result:   
| x  | y  | z  | triangle |   
|----|----|----|----------|   
| 13 | 15 | 30 | No       |   
| 10 | 20 | 15 | Yes      |   
  
\*/*   
*-- condition to form a triangle is a+b > c and a+c > b and b+c > a*   
SELECT x,   
       y,   
       z,   
       CASE   
         WHEN x + y > z   
              AND x + z > y   
              AND y + z > x THEN 'Yes'   
         ELSE 'No'   
       END AS triangle   
FROM   triangle   
  
*/\*   
601. Human Traffic of Stadium   
X city built a new stadium, each day many people visit it and the stats are saved as these columns: id, date, people   
  
Please write a query to display the records which have 3 or more consecutive rows and the amount of people more than 100(inclusive).   
  
For example, the table stadium:   
+------+------------+-----------+   
| id   | date       | people    |   
+------+------------+-----------+   
| 1    | 2017-01-01 | 10        |   
| 2    | 2017-01-02 | 109       |   
| 3    | 2017-01-03 | 150       |   
| 4    | 2017-01-04 | 99        |   
| 5    | 2017-01-05 | 145       |   
| 6    | 2017-01-06 | 1455      |   
| 7    | 2017-01-07 | 199       |   
| 8    | 2017-01-08 | 188       |   
+------+------------+-----------+   
For the sample data above, the output is:   
  
+------+------------+-----------+   
| id   | date       | people    |   
+------+------------+-----------+   
| 5    | 2017-01-05 | 145       |   
| 6    | 2017-01-06 | 1455      |   
| 7    | 2017-01-07 | 199       |   
| 8    | 2017-01-08 | 188       |   
+------+------------+-----------+   
Note:   
Each day only have one row record, and the dates are increasing with id increasing.   
\*/*   
SELECT DISTINCT s1.id,   
                s1.date,   
                s1.people   
FROM   stadium s1,   
       stadium s2,   
       stadium s3   
WHERE  s1.people >= 100   
       AND s2.people >= 100   
       AND s3.people >= 100   
       AND ( ( s1.id = s2.id - 1   
               AND s2.id = s3.id - 1   
               AND s1.id = s3.id - 2 )   
              OR ( s2.id = s1.id - 1   
                   AND s1.id = s3.id - 1   
                   AND s2.id = s3.id - 2 )   
              OR ( s3.id = s2.id - 1   
                   AND s2.id = s1.id - 1   
                   AND s3.id = s1.id - 2 ) )   
ORDER  BY s1.id   
  
*/\*   
579. Find Cumulative Salary of an Employee   
The Employee table holds the salary information in a year.   
  
Write a SQL to get the cumulative sum of an employee's salary over a period of 3 months but exclude the most recent month.   
  
The result should be displayed by 'Id' ascending, and then by 'Month' descending.   
  
Example   
Input   
  
| Id | Month | Salary |   
|----|-------|--------|   
| 1  | 1     | 20     |   
| 2  | 1     | 20     |   
| 1  | 2     | 30     |   
| 2  | 2     | 30     |   
| 3  | 2     | 40     |   
| 1  | 3     | 40     |   
| 3  | 3     | 60     |   
| 1  | 4     | 60     |   
| 3  | 4     | 70     |   
Output   
| Id | Month | Salary |   
|----|-------|--------|   
| 1  | 3     | 90     |   
| 1  | 2     | 50     |   
| 1  | 1     | 20     |   
| 2  | 1     | 20     |   
| 3  | 3     | 100    |   
| 3  | 2     | 40     |   
Explanation   
Employee '1' has 3 salary records for the following 3 months except the most recent month '4': salary 40 for month '3', 30 for month '2' and 20 for month '1'   
So the cumulative sum of salary of this employee over 3 months is 90(40+30+20), 50(30+20) and 20 respectively.   
  
| Id | Month | Salary |   
|----|-------|--------|   
| 1  | 3     | 90     |   
| 1  | 2     | 50     |   
| 1  | 1     | 20     |   
Employee '2' only has one salary record (month '1') except its most recent month '2'.   
| Id | Month | Salary |   
|----|-------|--------|   
| 2  | 1     | 20     |   
Employ '3' has two salary records except its most recent pay month '4': month '3' with 60 and month '2' with 40. So the cumulative salary is as following.   
| Id | Month | Salary |   
|----|-------|--------|   
| 3  | 3     | 100    |   
| 3  | 2     | 40     |   
\*/*   
SELECT id,   
       month,   
       cum\_salary salary   
FROM   (SELECT id,   
               month,   
               salary,   
               (SELECT *Sum*(salary)   
                FROM   employee e2   
                WHERE  e2.id = e11.id   
                       AND e2.month <= e11.month) cum\_salary   
        FROM   (SELECT id,   
                       month,   
                       salary,   
                       (SELECT *Max*(month)   
                        FROM   employee e3   
                        WHERE  e3.id = e1.id) max\_month   
                FROM   employee e1) e11   
        WHERE  month <> max\_month) t3   
ORDER  BY id,   
          month DESC   
  
*/\*   
615. Average Salary: Departments VS Company   
Given two tables as below, write a query to display the comparison result (higher/lower/same) of the average salary of employees in a department to the company's average salary.   
  
Table: salary   
| id | employee\_id | amount | pay\_date   |   
|----|-------------|--------|------------|   
| 1  | 1           | 9000   | 2017-03-31 |   
| 2  | 2           | 6000   | 2017-03-31 |   
| 3  | 3           | 10000  | 2017-03-31 |   
| 4  | 1           | 7000   | 2017-02-28 |   
| 5  | 2           | 6000   | 2017-02-28 |   
| 6  | 3           | 8000   | 2017-02-28 |   
The employee\_id column refers to the employee\_id in the following table employee.   
| employee\_id | department\_id |   
|-------------|---------------|   
| 1           | 1             |   
| 2           | 2             |   
| 3           | 2             |   
So for the sample data above, the result is:   
| pay\_month | department\_id | comparison  |   
|-----------|---------------|-------------|   
| 2017-03   | 1             | higher      |   
| 2017-03   | 2             | lower       |   
| 2017-02   | 1             | same        |   
| 2017-02   | 2             | same        |   
Explanation   
In March, the company's average salary is (9000+6000+10000)/3 = 8333.33...   
The average salary for department '1' is 9000, which is the salary of employee\_id '1' since there is only one employee in this department. So the comparison result is 'higher' since 9000 > 8333.33 obviously.   
The average salary of department '2' is (6000 + 10000)/2 = 8000, which is the average of employee\_id '2' and '3'. So the comparison result is 'lower' since 8000 < 8333.33.   
With he same formula for the average salary comparison in February, the result is 'same' since both the department '1' and '2' have the same average salary with the company, which is 7000.   
  
\*/*   
SELECT dept\_avg.sal\_month AS pay\_month,   
       dept\_avg.department\_id,   
       CASE   
         WHEN dept\_avg.avg\_sal > comp\_avg.avg\_sal THEN 'higher'   
         WHEN dept\_avg.avg\_sal < comp\_avg.avg\_sal THEN 'lower'   
         ELSE 'same'   
       END                AS comparison   
FROM   (SELECT e.department\_id,   
               **Date\_format**(pay\_date, '%Y-%m') sal\_month,   
               *Avg*(s.amount)                  avg\_sal   
        FROM   employee e,   
               salary s   
        WHERE  e.employee\_id = s.employee\_id   
        GROUP  BY e.department\_id,   
                  **Date\_format**(pay\_date, '%Y-%m')) dept\_avg,   
       (SELECT **Date\_format**(pay\_date, '%Y-%m') sal\_month,   
               *Avg*(s.amount)                  avg\_sal   
        FROM   salary s   
        GROUP  BY **Date\_format**(pay\_date, '%Y-%m')) comp\_avg   
WHERE  dept\_avg.sal\_month = comp\_avg.sal\_month   
  
*/\*   
603. Consecutive Available Seats   
Several friends at a cinema ticket office would like to reserve consecutive available seats.   
Can you help to query all the consecutive available seats order by the seat\_id using the following cinema table?   
  
| seat\_id | free |   
|---------|------|   
| 1       | 1    |   
| 2       | 0    |   
| 3       | 1    |   
| 4       | 1    |   
| 5       | 1    |   
Your query should return the following result for the sample case above.   
| seat\_id |   
|---------|   
| 3       |   
| 4       |   
| 5       |   
Note:   
The seat\_id is an auto increment int, and free is bool ('1' means free, and '0' means occupied.).   
Consecutive available seats are more than 2(inclusive) seats consecutively available.   
\*/*   
SELECT DISTINCT c1.seat\_id seat\_id   
FROM   cinema c1,   
       cinema c2   
WHERE  ( c2.seat\_id = c1.seat\_id + 1   
          OR c1.seat\_id = c2.seat\_id + 1 )   
       AND c1.free = 1   
       AND c2.free = 1   
ORDER  BY seat\_id