**177. Nth Highest Salary**

Medium

216169FavoriteShare

Write a SQL query to get the *n*th highest salary from the Employee table.

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

| Id | Salary |

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

| 1 | 100 |

| 2 | 200 |

| 3 | 300 |

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

For example, given the above Employee table, the *n*th highest salary where *n* = 2 is 200. If there is no *n*th highest salary, then the query should return null.

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

| getNthHighestSalary(2) |

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

| 200 |

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

**My solution: very slow, not recommended**

**Mysql use “ # “ for comments while MS sql server use “ /\* \*/ “ for comments**

CREATE FUNCTION getNthHighestSalary(@N INT) RETURNS INT # @ is a must be

BEGIN

RETURN (

/\* Write your T-SQL query statement below. \*/

SELECT Salary # the final output is salary

FROM

(SELECT Salary,

ROW\_NUMBER() OVER(ORDER BY Salary DESC) AS row\_num # desc because nth highest salary

FROM Employee

GROUP BY Salary) T1 # group salary of same amount together

WHERE T1.row\_num = @N

);

END

**@ to define user variable N**

**Row\_number() is a window function. Some mySQL version may not support. Can run in MySQL8.**

**Change to MS sql server, it would work on leetcode.**

<http://www.mysqltutorial.org/mysql-window-functions/mysql-row_number-function/>

tutorials for row\_num and examples

**# method 1: (select , group by )**

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

**SET N = N - 1;**

**RETURN (**

**SELECT Salary**

**FROM Employee**

**GROUP BY Salary**

**ORDER BY Salary DESC**

**LIMIT 1 OFFSET N**

**);**

END

**# method 2: use SELECT DISTINCT (donot need group by, it would return null )**

CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT

BEGIN

**SET N = N - 1;**

**RETURN (**

# Write your MySQL query statement below.

**SELECT DISTINCT Salary**

**FROM Employee**

**ORDER BY Salary DESC**

**LIMIT N, 1** # LIMIT 1 OFFSET m

**);**

**END**

**Note**

* SELECT DISTINCT is to avoid duplication
* When only one expression is provided in the DISTINCT clause, the query will return the unique values for that expression.
* When more than one expression is provided in the DISTINCT clause, the query will retrieve unique combinations for the expressions listed.
* **In SQL**, the DISTINCT clause doesn't ignore NULL values. So when using the DISTINCT clause in your SQL statement, your result set **will include NULL as a distinct value.**

**184. Department Highest Salary**

Medium

26568FavoriteShare

SQL Schema

The Employee table holds all employees. Every employee has an Id, a salary, and there is also a column for the department Id.

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

| Id | Name | Salary | DepartmentId |

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

| 1 | Joe | 70000 | 1 |

| 2  | Jim   | 90000  | 1            |

| 3 | Henry | 80000 | 2 |

| 4 | Sam | 60000 | 2 |

| 5 | Max | 90000 | 1 |

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

The Department table holds all departments of the company.

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

| Id | Name |

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

| 1 | IT |

| 2 | Sales |

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

Write a SQL query to find employees who have the highest salary in each of the departments. For the above tables, your SQL query should return the following rows (order of rows does not matter).

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

| Department | Employee | Salary |

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

| IT | Max | 90000 |

| IT         | Jim      | 90000  |

| Sales | Henry | 80000 |

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

**Explanation:**

Max and Jim both have the highest salary in the IT department and Henry has the highest salary in the Sales department.

Explanation: subquery use limit, only return one row. We need highest salary in each department, so one department may have multiple rows.

Inner query should generate the highest salary of each department ( which is a list ) , outside query set the condition in the highest salary list

**My solution: First attempt :**

SELECT D.Name AS Department,

E.Name AS Employee,

E.Salary

FROM Employee E

JOIN Department D

ON E.DepartmentId = D.Id

WHERE E.Salary IN # this condition is not enough, salary = max\_slary, but may be from a different department

(SELECT MAX(Salary) AS max\_salary

FROM Employee E

GROUP BY DepartmentId);

**Correct solution: # two conditions to filter : id and salary**

**METHOD 1: set 2 conditions, both DepartmentId and Salary should match**

SELECT D.Name AS Department,

E1.Name AS Employee,

E1.Salary

FROM Employee E1

JOIN Department D

ON E1.DepartmentId = D.Id

**WHERE (E1.DepartmentId , E1.Salary) IN # meet these two conditions**

**(SELECT E2.DepartmentId, MAX(E2.Salary) AS max\_salary**

FROM Employee E2

GROUP BY E2.DepartmentId);

**Method 2 : put one of the 2 conditions in sub query**

SELECT Department.Name AS Department, E1.Name AS Employee, E1.Salary AS Salary

FROM Employee AS E1

JOIN Department

ON E1.DepartmentId = Department.Id

WHERE E1.Salary IN # both “ in “ and “ = ” work

(SELECT MAX(Salary)

FROM Employee AS E2

WHERE E1.DepartmentId = E2.DepartmentId)

**178. Rank Scores**

Medium

SQL Schema

Write a SQL query to rank scores. If there is a tie between two scores, both should have the same ranking. Note that after a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no "holes" between ranks.

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

| Id | Score |

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

| 1 | 3.50 |

| 2 | 3.65 |

| 3 | 4.00 |

| 4 | 3.85 |

| 5 | 4.00 |

| 6 | 3.65 |

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

For example, given the above Scores table, your query should generate the following report (order by highest score):

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

| Score | Rank |

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

| 4.00 | 1 |

| 4.00 | 1 |

| 3.85 | 2 |

| 3.65 | 3 |

| 3.65 | 3 |

| 3.50 | 4 |

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

Notes: leetcode mySQL does not support window function

**# method 1:**

# select two : all score from original table and the count of all distinct scores

# distinct scores would be a second table

SELECT A.Score, count(B.Score) as Rank

FROM Scores as A,

(SELECT DISTINCT Score

FROM Scores) as B

WHERE A.Score <= B.Score

# know how many distinct scores below this row's score in table "Score" , so we compare them and got a number, the number is rank.

GROUP BY A.Id

ORDER BY A.Score desc;

**# method 2: least time best solution**

**Explanation: two inner queries generated two derived tables: tmp and Rank ,**

**Second derived table RANK will be selected by the outer query**

SELECT Score,

(SELECT COUNT(\*)

FROM (SELECT DISTINCT Score AS Score2

FROM Scores) **tmp**

WHERE Score2 >= Score) **Rank**

FROM Scores

ORDER BY Score DESC;

**185. Department Top Three Salaries**

Hard

30974FavoriteShare

SQL Schema

The Employee table holds all employees. Every employee has an Id, and there is also a column for the department Id.

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

| Id | Name | Salary | DepartmentId |

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

| 1 | Joe | 85000 | 1 |

| 2 | Henry | 80000 | 2 |

| 3 | Sam | 60000 | 2 |

| 4 | Max | 90000 | 1 |

| 5 | Janet | 69000 | 1 |

| 6 | Randy | 85000 | 1 |

| 7 | Will | 70000 | 1 |

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

The Department table holds all departments of the company.

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

| Id | Name |

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

| 1 | IT |

| 2 | Sales |

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

Write a SQL query to find employees who earn the top three salaries in each of the department. For the above tables, your SQL query should return the following rows (order of rows does not matter).

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

| Department | Employee | Salary |

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

| IT | Max | 90000 |

| IT | Randy | 85000 |

| IT | Joe | 85000 |

| IT | Will | 70000 |

| Sales | Henry | 80000 |

| Sales | Sam | 60000 |

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

**Explanation:**

In IT department, Max earns the highest salary, both Randy and Joe earn the second highest salary, and Will earns the third highest salary. There are only two employees in the Sales department, Henry earns the highest salary while Sam earns the second highest salary.

SELECT D.Name AS Department,

E1.Name AS Employee,

E1.Salary

FROM Employee E1

JOIN Department D

ON D.Id = E1. DepartmentId

WHERE (SELECT COUNT(DISTINCT E2.Salary)

FROM Employee E2

WHERE E1.DepartmentId = E2.DepartmentId AND E1.Salary <= E2.Salary) <=3

ORDER BY D.Id ASC, E1.Salary DESC;

# same department will be apart if just order by salary

# order by D.Id function like a group by DepartmentId

**# less than and equal to <= , = sign always on the right**

**180. Consecutive Numbers**

Medium

21261FavoriteShare

SQL Schema

Write a SQL query to find all numbers that appear at least three times consecutively.

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

| Id | Num |

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

| 1 | 1 |

| 2 | 1 |

| 3 | 1 |

| 4 | 2 |

| 5 | 1 |

| 6 | 2 |

| 7 | 2 |

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

For example, given the above Logs table, 1 is the only number that appears consecutively for at least three times.

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

| ConsecutiveNums |

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

| 1 |

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

**# Method 1 ： 2 joins , but the limitation is the the IDs can not have gap , must be like 1, 2, 3……..**

Select DISTINCT l1.Num AS ConsecutiveNums

FROM Logs l1, Logs l2, Logs l3

WHERE l1.Id=l2.Id-1 AND l2.Id=l3.Id-1

AND l1.Num=l2.Num AND l2.Num=l3.Num

**# Method 2: 1 join**

select distinct l.Num as ConsecutiveNums

from Logs l , Logs r

where l.Num = r.Num and r.Id < l.Id and l.Id -r.Id <3

group by l.Num, l.Id

having sum(l.Id-r.Id)=(0+1+2)

# -- if you want at least four times, here is 6 = (0+1+2+3)

**626. Exchange Seats**

Medium

152149FavoriteShare

SQL Schema

Mary is a teacher in a middle school and she has a table seat storing students' names and their corresponding seat ids.

The column **id** is continuous increment.

Mary wants to change seats for the adjacent students.

Can you write a SQL query to output the result for Mary?

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

| id | student |

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

| 1 | Abbot |

| 2 | Doris |

| 3 | Emerson |

| 4 | Green |

| 5 | Jeames |

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

For the sample input, the output is:

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

| id | student |

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

| 1 | Doris |

| 2 | Abbot |

| 3 | Green |

| 4 | Emerson |

| 5 | Jeames |

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

**Note:**  
If the number of students is odd, there is no need to change the last one's seat.

**# explanation : just swap the seat ids for adjacent students, even number turns to add number, add number turns to even number. If last number is even, it does not have any number to swap, so it stays the same**

**My solution: wrong attempt : pay attention to the output , id should be first, the student name**

**Id = id , id = id + 1 , id = id -1 wrong**

SELECT

CASE

WHEN id = (SELECT COUNT(\*) FROM seat) AND id % 2 <> 0 THEN id = id

WHEN id < (SELECT COUNT(\*) FROM seat) AND id % 2 <> 0 THEN id = id +1

ELSE id = id - 1

END AS id , # do not forget the comma

student

FROM seat

ORDER BY id ;

**Correct solution:**

SELECT

CASE

WHEN id = (SELECT COUNT(\*) FROM seat) AND id % 2 <> 0 THEN id

WHEN id < (SELECT COUNT(\*) FROM seat) AND id % 2 <> 0 THEN id +1

ELSE id – 1 # else means id < (SELECT COUNT(\*) FROM seat) AND id % 2 = 0

END AS id , # do not forget the comma

student

FROM seat

ORDER BY id ;

Hard

**262. Trips and Users**

Hard

178144FavoriteShare

SQL Schema

The Trips table holds all taxi trips. Each trip has a unique Id, while Client\_Id and Driver\_Id are both foreign keys to the Users\_Id at the Users table. Status is an ENUM type of (‘completed’, ‘cancelled\_by\_driver’, ‘cancelled\_by\_client’).

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

| Id | Client\_Id | Driver\_Id | City\_Id | Status |Request\_at|

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

| 1 | 1 | 10 | 1 | completed |2013-10-01|

| 2 | 2 | 11 | 1 | cancelled\_by\_driver|2013-10-01|

| 3 | 3 | 12 | 6 | completed |2013-10-01|

| 4 | 4 | 13 | 6 | cancelled\_by\_client|2013-10-01|

| 5 | 1 | 10 | 1 | completed |2013-10-02|

| 6 | 2 | 11 | 6 | completed |2013-10-02|

| 7 | 3 | 12 | 6 | completed |2013-10-02|

| 8 | 2 | 12 | 12 | completed |2013-10-03|

| 9 | 3 | 10 | 12 | completed |2013-10-03|

| 10 | 4 | 13 | 12 | cancelled\_by\_driver|2013-10-03|

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

The Users table holds all users. Each user has an unique Users\_Id, and Role is an ENUM type of (‘client’, ‘driver’, ‘partner’).

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

| Users\_Id | Banned | Role |

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

| 1 | No | client |

| 2 | Yes | client |

| 3 | No | client |

| 4 | No | client |

| 10 | No | driver |

| 11 | No | driver |

| 12 | No | driver |

| 13 | No | driver |

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

Write a SQL query to find the cancellation rate of requests made by unbanned users between **Oct 1, 2013** and **Oct 3, 2013**. For the above tables, your SQL query should return the following rows with the cancellation rate being rounded to two decimal places.

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

| Day | Cancellation Rate |

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

| 2013-10-01 | 0.33 |

| 2013-10-02 | 0.00 |

| 2013-10-03 | 0.50 |

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

**Credits:**  
Special thanks to [@cak1erlizhou](https://leetcode.com/discuss/user/cak1erlizhou) for contributing this question, writing the problem description and adding part of the test cases.

# An **ENUM** is a string object with a value chosen from a list of permitted values that are **enumerated** explicitly in the column specification at table creation time

**# Explanation:**

**# two counts, one is for cancellation, one is for total**

**# there is no need for join. Just set the condition in WHERE clause**

SELECT T.Request\_at AS Day,

ROUND(COUNT(IF(T.Status != 'completed', TRUE, NULL)) / COUNT(\*) , 2) AS "Cancellation Rate"

FROM Trips T

WHERE (T.Request\_at BETWEEN '2013-10-01' AND '2013-10-03')

AND T.Client\_Id NOT IN (SELECT U.Users\_Id FROM Users U WHERE U.Banned = 'Yes')

AND T.Driver\_Id NOT IN (SELECT U.Users\_Id FROM Users U WHERE U.Banned = 'Yes')

GROUP BY Day;

**Notes:**

**# must use quotes for the column names with space**

**# 'between and' & 'not in' do not need 'is'**

**# use parenthesis for each condition , 3 conditions**

**# not just 'T.Client\_Id' but also 'T.Driver\_Id ' should not be banned**

**# according to the output, we need a GROUP BY Clause**

**# in mySQL , just use date like** '2013-10-01' directly , do not need date\_trunc()