**SQL Questions:**

1. **Determine the nth or 5th highest salary**

SELECT TOP 1 Salary FROM (

SELECT DISTINCT TOP 4 Salary FROM employee

ORDER BY Salary DESC ) a

ORDER BY Salary

1. **Determine the 5th highest salary without using TOP or limit method.**

SELECT e1.Salary FROM Employee e1

WHERE (N -1) = ( SELECT COUNT(DISTINCT(Salary)) FROM Employee e2

WHERE e1.salary < e2.salary );

|  |  |
| --- | --- |
| ID | Salary |
| 1 | 100 |

The above solution might not work if there is just one record say,

SELECT MAX(salary) FROM (

SELECT \*, DENSE\_RANK() OVER (ORDER BY salary DESC) salary\_rank

FROM employee

) T

WHERE salary\_rank = @N

1. **https://leetcode.com/problems/nth-highest-salary/submissions/**

CREATE FUNCTION getNthHighestSalary(@N INT) RETURNS INT AS

BEGIN

RETURN (

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

SELECT max(Salary) FROM (

select \*, dense\_rank() over (order by Salary desc) salary\_rank

FROM Employee

) t

where salary\_rank = @N

);

END

1. **Find employees who earn the top N salaries**

SELECT e1.Salary, e1.name FROM Employee e1

WHERE N > ( SELECT COUNT(DISTINCT(Salary)) FROM Employee e2

WHERE e1.salary < e2.salary)

1. **find employees who earn the top three salaries in each of the department.**

SELECT d.Name AS 'Department', e1.Name AS 'Employee', e1.Salary

FROM Employee e1 JOIN Department d

ON e1.DepartmentId = d.Id

WHERE 3 > (SELECT COUNT(DISTINCT e2.Salary

FROM Employee e2

WHERE e1.Salary< e2.Salary

AND e1.DepartmentId = e2.DepartmentId );

1. **Write an SQL query to show one row twice in results from a table.**

Select \* from Employee e1

UNION ALL

Select \* from Employee e2

1. **Write an SQL query to fetch the list of employees with the same salary.**

SELECT e1.name, e2.name, e1.salary FROM employee e1, Employee e2

WHERE e1.salary = e2.salary AND e1.name != e2.name

1. **Facebook Question**

|  |
| --- |
|  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | sms\_sends` table   | **column** | **type** | | --- | --- | | ds | datetime | | country | string | | carrier | string | | phone\_number | integer | | type | string |   confirmers` table   | **column** | **type** | | --- | --- | | date | datetime | | phone\_number | integer |   We're given two tables, one that represents SMS sends to phone numbers and the other represents confirmations from SMS sends.  **A) Write a query to calculate the number of phone numbers that we sent 'confirmation' SMS texts to yesterday by carrier and country.**  **Example:**   | **ds** | **country** | **carrier** | **phone\_number** | **type** | | --- | --- | --- | --- | --- | | 2017-10-01 | US | sprint | 8005551111 | confirmation | | 2017-10-01 | US | sprint | 8005551111 | friend\_request | | 2017-10-01 | US | at&t | 8005550000 | confirmation | | 2017-10-01 | US | at&t | 8005550000 | confirmation | | 2017-10-01 | CA | rogers | 8005550101 | message |   SELECT carrier, country, COUNT(DISTINCT phone\_number)  FROM sms\_sends WHERE type = ‘confirmation’  AND cast(ds as date) = dateadd(day,-1, cast(getdate() as date))  GROUP BY carrier, country;  **B) Calculate what percentage of users were confirmed per day.**  Select COUNT(DISTINCT(c.phone\_number)/COUNT(DISTINCT(s.phone\_number))  FROM sms\_sends s LEFT JOIN confirmers c ON c.phone\_number = s.phone\_number AND c.date = s.ds  WHERE type = ‘confirmation’ GROUP BY date;   1. **employees table**  | **columns** | **types** | | --- | --- | | id | int | | first\_name | varchar | | last\_name | varchar | | salary | int | | department\_id | int |   **departments table**   | **columns** | **types** | | --- | --- | | id | int | | name | varchar |   **Given the tables above, select the top 3 departments by the highest percentage of employees making over 100K in salary and have at least 10 employees.**  **Example output:**   | **> 100K %** | **department name** | **number of employees** | | --- | --- | --- | | 90% | engineering | 25 | | 50% | marketing | 50 | | 12% | sales | 12 |   SELECT d.name, SUM(CASE WHEN e.salary > 100000 THEN 1 ELSE 0 END)/COUNT(DISTINCT e.id)AS pct\_above\_100k, COUNT(DISTINCT e.id) AS c  FROM employees e JOIN departments d ON e.department\_id = d.id  GROUP BY 1 HAVING COUNT(\*) > 10 ORDER BY 2 DESC LIMIT 3   1. **Each row in the table represents an individual user product purchase.**   transactions` table   | column | type | | --- | --- | | id | integer | | user\_id | integer | | created\_at | datetime | | product\_id | integer | | quantity | integer |   Write a query to get the number of customers that were upsold by purchasing additional products.  **Note** that if the customer purchased two things on the same day that does not count as an upsell as they were purchased within a similar timeframe.   1. **Given a table salary, such as the one below, that has m=male and f=female values. Swap all f and m values (i.e., change all f values to m and vice versa) with a single update statement and no intermediate temp table.**   UPDATE salary SET sex =  CASE sex WHEN 'f' then 'm'  WHEN 'm' then 'f'  ELSE sex -- just in case it contains a new value, we keep it  END;  OR  UPDATE salary  SET sex =  CASE  when sex = 'f' then 'm'  when sex = 'm' then 'f'  else sex END; | |

1. **Please write a SQL query to output movies with an odd numbered ID and a description that is not 'boring'. Order the result by rating.**

SELECT id,movie,description,rating from cinema

where id%2 != 0 AND description NOT LIKE '%boring%' ORDER BY rating DESC;

1. **Write a SQL query to find all duplicate emails in a table named Person**

| Id | Email |

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

| 1 | a@b.com |

| 2 | c@d.com |

| 3 | [a@b.com](mailto:a@b.com)

Select Distinct p1.email from Person p1, Person p2

Where p1.id != p2.id and p1.email = p2.email

1. **Write a SQL query to**delete**all duplicate email entries in a table named Person, keeping only unique emails based on its smallest Id**

DELETE FROM Person WHERE Id IN (

SELECT p.Id

FROM (

SELECT Id,

Email,

ROW\_NUMBER() OVER(PARTITION BY Email ORDER BY Id) AS email\_duplicate\_position

FROM Person

) p

WHERE p.email\_duplicate\_position > 1

);

1. **Write an SQL query to find all dates' id with higher temperature compared to its previous dates (yesterday).**

select b.id from Weather a, Weather b

where b.Recorddate = dateadd(day,1,a.recorddate)

and a.temperature < b.temperature

OR

select id

from (

select recorddate, lag(recorddate, 1) over(order by recorddate) as lag\_date, id,Temperature,

lag(temperature, 1) over(order by recorddate) as lag\_temp

from weather

) w

where w.temperature > w.lag\_temp

and datediff(day, w.lag\_date, w.recorddate ) = 1

1. **write a SQL query that finds out employees who earn more than their managers**

<https://leetcode.com/problems/employees-earning-more-than-their-managers/solution/>

Select a.name from Employee a, Employee b

WHERE a.managerID = b.ID

AND a.Salary>b.salary

OR

SELECT a.NAME AS Employee

FROM Employee AS a JOIN Employee AS b

ON a.ManagerId = b.Id

AND a.Salary > b.Salary

1. **Mary wants to change seats for the adjacent students. If the number of students is odd, there is no need to change the last one's seat.**

id | student

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

| 1 | Abbot |

| 2 | Doris |

| 3 | Emerson |

| 4 | Green |

| 5 | Jeames

id | student |

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

| 1 | Doris |

| 2 | Abbot |

| 3 | Green |

| 4 | Emerson |

| 5 | Jeames

SELECT

(CASE

WHEN (id%2) != 0 AND counts != id THEN id + 1

WHEN (id% 2) != 0 AND counts = id THEN id

ELSE id - 1

END) AS id,

student

FROM

seat,

(SELECT

COUNT(\*) AS counts

FROM

seat) AS seat\_counts

ORDER BY id ASC;

**OR**

SELECT id,

CASE

WHEN (id%2) != 0 AND next\_student IS NOT NULL THEN next\_student

WHEN (id%2) = 0 THEN prev\_student

ELSE student

END AS student

FROM (

SELECT id,

student,

LAG(student, 1) OVER (order by id) AS prev\_student,

LEAD(student, 1) OVER (order by id) AS next\_student

FROM seat

) a

1. **Write a SQL query to rank score**

Id | Score |

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

| 1 | 3.50 |

| 2 | 3.65 |

| 3 | 4.00 |

| 4 | 3.85 |

| 5 | 4.00 |

| 6 | 3.65

Select Score,

DENSE\_RANK() OVER (ORDER BY Score DESC) as Rank

FROM Scores

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

Id | Num |

|  |  |
| --- | --- |
| Id | Num |
| 1 | 3 |
| 2 | 3 |
| 3 | 3 |
| 4 | 3 |

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

| 1 | 1 |

| 2 | 1 |

| 3 | 1 |

| 4 | 2 |

| 5 | 1 |

| 6 | 2 |

| 7 | 2

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

OR

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 ;

1. **find employees who have the highest salary in each of the departments**

Id | Name | Salary | DepartmentId |

|  |  |
| --- | --- |
| ID | Name |
| 1 | IT |
| 2 | Sales |

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

| 1 | Joe | 70000 | 1 |

| 2  | Jim   | 90000  | 1            |

| 3 | Henry | 80000 | 2 |

| 4 | Sam | 60000 | 2 |

| 5 | Max | 90000 | 1

SELECT d.Name as Department, e.Name as Employee, e.Salary

FROM Employee as e

JOIN Department as d

on e.DepartmentId = d.Id

JOIN (

SELECT MAX(Salary) as Salary, DepartmentId

FROM Employee

GROUP BY DepartmentId

) as mx

ON e.Salary = mx.Salary AND e.DepartmentId = mx.DepartmentId;

1. [**https://leetcode.com/problems/human-traffic-of-stadium/**](https://leetcode.com/problems/human-traffic-of-stadium/)

-------This is good example for cross join. We are selecting t1.\* so we have to list all three combinations where t1 is in the beginning , t1 in the middle and t1 at the end. Otherwise, t1.\* would give us only one combination say 5, 6, 7 and it would skip (6, 7, 8). Using OR condition would get both the combinations.

SELECT distinct t1.\*

FROM stadium t1, stadium t2, stadium t3

WHERE t1.people >= 100 AND t2.people >= 100 AND t3.people >= 100

AND

(

(t1.id - t2.id = -1 AND t1.id - t3.id =-2 AND t2.id - t3.id =-1)

OR

(t2.id - t1.id = -1 AND t2.id - t3.id = -2 AND t1.id - t3.id =-1)

OR

(t3.id - t2.id = -1 AND t2.id - t1.id =-1 AND t3.id - t1.id = -2)

)

ORDER BY t1.id ;

1. [h**ttps://leetcode.com/problems/department-top-three-salaries/**](https://leetcode.com/problems/department-top-three-salaries/)

SELECT d.Name AS 'Department', e1.Name AS 'Employee', e1.Salary

FROM Employee e1 JOIN Department d

ON e1.DepartmentId = d.Id

WHERE 3 > (

SELECT COUNT(DISTINCT e2.Salary)

FROM Employee e2

WHERE e1.Salary< e2.Salary

AND e1.DepartmentId = e2.DepartmentId

);

1. [**https://leetcode.com/problems/trips-and-users/**](https://leetcode.com/problems/trips-and-users/)

WITH cl AS (

SELECT \* FROM users

WHERE banned='No'

AND role='client'),

--Get the drivers list

dr AS (

SELECT \* FROM users

WHERE banned='No'

AND role='driver'),

--Get the raw data as per filters

data AS(

SELECT \* FROM trips

WHERE client\_id IN (SELECT users\_id FROM cl)

AND driver\_id IN (SELECT users\_id FROM dr)

AND request\_at BETWEEN '2013-10-01' AND '2013-10-03')

SELECT request\_at day, ROUND(SUM(CASE

WHEN status!='completed' THEN 1

ELSE 0

END)\*1.00/COUNT(id) ,2) "Cancellation Rate"

FROM data

GROUP BY request\_at