--Classes More Than 5 Students

--There is a table courses with columns: student and class

--Please list out all classes which have more than or equal to 5 students.

--For example, the table:

--

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

--| student | class |

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

--| A | Math |

--| B | English |

--| C | Math |

--| D | Biology |

--| E | Math |

--| F | Computer |

--| G | Math |

--| H | Math |

--| I | Math |

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

--Should output:

--

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

--| class |

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

--| Math |

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

CREATE TABLE Class\_HW (student VARCHAR(30), class VARCHAR(30));

INSERT INTO Class\_HW

VALUES

('A', 'Math'),

('B', 'English'),

('C', 'Math'),

('D', 'Biology'),

('E', 'Math'),

('F', 'Computer'),

('G', 'Math'),

('H', 'Math'),

('I', 'Math');

SELECT class

FROM Class\_HW

GROUP BY class

HAVING count(\*) >= 5;

SELECT a.class

FROM

(SELECT class, count(\*) as c

FROM Class\_HW

GROUP BY class) a

WHERE a.c >= 5;

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

-- https://leetcode.com/problems/duplicate-emails/

-- input

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

--| Id | Email |

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

--| 1 | a@b.com |

--| 2 | c@d.com |

--| 3 | a@b.com |

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

-- output

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

--| Email |

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

--| a@b.com |

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

CREATE TABLE Email\_HW1 (Id INT, Email VARCHAR(30));

INSERT INTO Email\_HW1

VALUES

(1, 'a@b.com'),

(2, 'c@d.com'),

(3, 'a@b.com');

SELECT Email

FROM Email\_HW1

GROUP BY Email

HAVING COUNT(\*) > 1;

SELECT a.Email

FROM

(SELECT Email, COUNT(\*) AS c

FROM Email\_HW1

GROUP BY Email) a

WHERE c > 1

;

-- Customers Who Never Order

-- https://leetcode.com/problems/customers-who-never-order/

--Suppose that a website contains two tables, the Customers table and the Orders table.

--Write a SQL query to find all customers who never order anything.

--Table: Customers.

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

--| Id | Name |

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

--| 1 | Joe |

--| 2 | Henry |

--| 3 | Sam |

--| 4 | Max |

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

--Table: Orders.

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

--| Id | CustomerId |

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

--| 1 | 3 |

--| 2 | 1 |

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

--Using the above tables as example, return the following:

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

--| Customers |

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

--| Henry |

--| Max |

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

CREATE TABLE Customers\_HW (Id INT, Name VARCHAR(30));

INSERT INTO Customers\_HW

VALUES

(1, 'Joe'),

(2, 'Henry'),

(3, 'Sam'),

(4, 'Max');

CREATE TABLE Orders\_HW (Id INT, CustomerId INT);

INSERT INTO Orders\_HW

VALUES

(1, 3),

(2, 1);

SELECT Customers\_HW.Name AS Customers

FROM (Customers\_HW LEFT JOIN Orders\_HW ON (Customers\_HW.ID = Orders\_HW.CustomerId))

WHERE Orders\_HW.CustomerId IS NULL

ORDER BY Customers\_HW.Name;

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

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

--| 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 |

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

CREATE TABLE Email\_HW (Id INT, Email VARCHAR(30));

INSERT INTO Email\_HW

VALUES

(1, 'john@example.com'),

(2, 'bob@example.com'),

(3, 'john@example.com');

-- SOLUTION 1

SELECT DISTINCT a.Email

FROM

(SELECT \* FROM Email\_HW) a

JOIN

(SELECT \* FROM Email\_HW) b

ON (a.Email = b.Email)

WHERE a.Id <= b.Id;

-- SOLUTION 2.

SELECT DISTINCT FIRST\_VALUE(Id) OVER(PARTITION BY Email ORDER BY Id ASC) AS Id,

Email

FROM Email\_HW

ORDER BY Id ASC;

-- find the Nth highest salary

CREATE TABLE Employee\_HW (Id INT, Salary INT);

INSERT INTO Employee\_HW

VALUES

(1, 1000),

(2, 2000),

(3, 3000),

(4, 4000),

(5, 5000);

-- N = 5

SELECT \*

FROM Employee\_HW

ORDER BY Salary DESC

LIMIT 4,1;

-- N = 2

SELECT \*

FROM Employee\_HW

ORDER BY Salary DESC

LIMIT 1,1;

-- find who has the most num of transaction records.

CREATE TABLE Transaction\_HW

(

id INT DEFAULT NULL,

num\_trans INT DEFAULT NULL,

dt VARCHAR(30) DEFAULT NULL

);

INSERT INTO Transaction\_HW (id, num\_trans, dt)

VALUES

(1, 3, '20190501'),

(1, 2, '20190502'),

(2, 1, '20190501'),

(2, 4, '20190502'),

(3, 2, '20190501')

;

-- should output

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

--| Id |

--| 1 |

--| 2 |

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

-- use a temp table

WITH tmp1 (id, count)

AS (

SELECT id, COUNT(\*) AS count FROM Transaction\_HW GROUP BY id

)

SELECT id

FROM tmp1

WHERE count = (SELECT max(count) FROM tmp1);

-- use nested query

SELECT b.id

FROM

(SELECT a.id as id, a.c AS c, max(a.c) OVER() AS m\_c

FROM

(SELECT id, count(\*) AS c FROM Transaction\_HW GROUP BY id) a) b

WHERE b.c = b.m\_c;

-- order by and then limit?? Will miss several users.

select id, sum(num\_trans) AS s from Transaction\_HW group by id order by s DESC;

-- what is wrong with the following query?

select id

from

(select id, count(\*) AS c from Transaction\_HW group by id) a

where c = (select max(c) from a);

--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?

-- ORIGINAL

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

--| id | student |

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

--| 1 | Abbot |

--| 2 | Doris |

--| 3 | Emerson |

--| 4 | Green |

--| 5 | Jeames |

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

-- NEW

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

--| id | student |

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

--| 1 | Doris |

--| 2 | Abbot |

--| 3 | Green |

--| 4 | Emerson |

--| 5 | Jeames |

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

CREATE TABLE Seat\_HW (Id INT, Student VARCHAR(30));

INSERT INTO Seat\_HW

VALUES

(1, 'Abbot'),

(2, 'Doris'),

(3, 'Emerson'),

(4, 'Green'),

(5, 'Jeames');

-- SOLUTION 1: use lead and lag

SELECT id, student, LEAD(student, 1, student) OVER(), LAG(student, 1, student) OVER()

FROM Seat\_HW;

SELECT a.id AS id,

CASE WHEN a.id % 2 = 1 THEN a.next

ELSE a.prev

END AS student

FROM

(SELECT id,

student,

LEAD(student, 1, student) OVER (ORDER BY id) AS next,

LAG(student, 1, student) OVER (ORDER BY id) AS prev

FROM Seat\_HW) a;

-- SOLUTION 2: modify the id

SELECT

(CASE

WHEN a.id%2 != 0 AND counts != a.id THEN a.id + 1

WHEN a.id%2 != 0 AND counts = a.id THEN a.id

ELSE a.id - 1

END) AS id,

a.student

FROM

(SELECT \*, COUNT(\*) OVER() AS counts

FROM Seat\_HW) AS a

ORDER BY id ASC;

-- table friend: tb1 = [user\_id , action, target\_id , date].

-- Action in {‘sent’, ‘accept’}. Calculate acceptance rate for all the requests sent in April.

-- Assumption: (a). each user can only send one request to a target user.

-- (b) each user can only accept the request from the same sender once.

-- (c) if A send request to B, B will have to accepted to be considered in our calculation.

select count(case when b.action = 'accept' then 1 end)/count(\*) as rate

from tb1 a left join tb1 b

on a.target\_id = b.user\_id and a.user\_id = b.target\_id

where a.action = 'send'

group by a.date

order by a.date;

-- we will need to create 2 tables.

-- table a: contains all the sent records.

-- table b: contains all the accept records.

-- sent target from a should be the users id from b.

-- and accept target from b should be the user id from a.

-- Groupon interview question:

-- For every user, select the most recent two transection.

-- Table=[user\_id, trans\_id, time]

SELECT a.user\_id, a.trans\_id, a.time

FROM

(SELECT user\_id, trans\_id, ROW\_NUMBER() OVER (PARTITION BY user\_id ORDER BY time DESC) AS rn

FROM tbl ) a

WHERE a.rn <= 2;

-- fb interview question:

2nd degree friends.

Given table users = [user1, user2]

Each row represents a first degree friend, how to find all the 2nd degree friends?

Output [user1, common\_friend, user2]

CREATE TABLE user\_fb\_HW (user1 VARCHAR(30), user2 VARCHAR(30));

INSERT INTO user\_fb\_HW

VALUES

('b', 'a'), ('a', 'c'), ('b', 'c'), ('d', 'c'), ('a', 'e'), ('f', 'e');

Select \*

FROM user\_fb\_HW a JOIN user\_fb\_HW b

ON a.user2 = b.user1;

SELECT \* FROM user\_fb\_HW; -- you will find out that users are not in order.

-- First step:

WITH ord\_tbl (user1, user2)

AS (

SELECT CASE WHEN user1 <= user2 THEN user1 ELSE user2 END AS user1,

CASE WHEN user1 <= user2 THEN user2 ELSE user1 END AS user2

FROM user\_fb\_HW

)

-- second step

-- does not include all records  
WITH ord\_tbl (user1, user2)

AS (

SELECT CASE WHEN user1 <= user2 THEN user1 ELSE user2 END AS user1,

CASE WHEN user1 <= user2 THEN user2 ELSE user1 END AS user2

FROM user\_fb\_HW

)

SELECT \*

FROM ord\_tbl a join ord\_tbl b ON a.user2 = b.user1;

-- third step

WITH ord\_tbl (user1, user2)

AS (

SELECT CASE WHEN user1 <= user2 THEN user1 ELSE user2 END AS user1,

CASE WHEN user1 <= user2 THEN user2 ELSE user1 END AS user2

FROM user\_fb\_HW

)

SELECT \*

FROM ord\_tbl a join ord\_tbl b ON

(a.user2 = b.user1 AND a.user1 != b.user2) OR

(a.user1 = b.user2 AND a.user2 != b.user1) OR

(a.user1 = b.user1 AND a.user2 != b.user2 ) OR

(a.user2 = b.user2 AND a.user1 != b.user1);

-- fourth trial

WITH ord\_tbl (user1, user2)

AS (

SELECT CASE WHEN user1 <= user2 THEN user1 ELSE user2 END AS user1,

CASE WHEN user1 <= user2 THEN user2 ELSE user1 END AS user2

FROM user\_fb\_HW

)

SELECT DISTINCT

CASE

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 <= b.user2 THEN a.user1

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 > b.user2 THEN b.user2

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 <= b.user1 THEN a.user2

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 > b.user1 THEN b.user1

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 <= b.user2 THEN a.user2

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 > b.user2 THEN b.user2

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 <= b.user1 THEN a.user1

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 > b.user1 THEN b.user1

END AS user1,

CASE

WHEN a.user2 = b.user1 AND a.user1 != b.user2 THEN a.user2

WHEN a.user1 = b.user2 AND a.user2 != b.user1 THEN a.user1

WHEN a.user1 = b.user1 AND a.user2 != b.user2 THEN a.user1

WHEN a.user2 = b.user2 AND a.user1 != b.user1 THEN a.user2

END AS common\_friend,

CASE

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 <= b.user2 THEN b.user2

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 > b.user2 THEN a.user1

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 <= b.user1 THEN b.user1

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 > b.user1 THEN a.user2

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 <= b.user2 THEN b.user2

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 > b.user2 THEN a.user2

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 <= b.user1 THEN b.user1

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 > b.user1 THEN a.user1

END AS user2

FROM ord\_tbl a join ord\_tbl b ON

(a.user2 = b.user1 AND a.user1 != b.user2) OR

(a.user1 = b.user2 AND a.user2 != b.user1) OR

(a.user1 = b.user1 AND a.user2 != b.user2) OR

(a.user2 = b.user2 AND a.user1 != b.user1);

SELECt \* FROM user\_fb\_HW;

-- notice that b and c are first degree friends…

-- need to join with ord\_tbl to remove first degree friends.

-- final solution:

WITH ord\_tbl (user1, user2)

AS (

SELECT CASE WHEN user1 <= user2 THEN user1 ELSE user2 END AS user1,

CASE WHEN user1 <= user2 THEN user2 ELSE user1 END AS user2

FROM user\_fb\_HW

),

second\_degree\_tbl (user1, common\_friend, user2)

AS(

SELECT DISTINCT

CASE

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 <= b.user2 THEN a.user1

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 > b.user2 THEN b.user2

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 <= b.user1 THEN a.user2

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 > b.user1 THEN b.user1

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 <= b.user2 THEN a.user2

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 > b.user2 THEN b.user2

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 <= b.user1 THEN a.user1

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 > b.user1 THEN b.user1

END AS user1,

CASE

WHEN a.user2 = b.user1 AND a.user1 != b.user2 THEN a.user2

WHEN a.user1 = b.user2 AND a.user2 != b.user1 THEN a.user1

WHEN a.user1 = b.user1 AND a.user2 != b.user2 THEN a.user1

WHEN a.user2 = b.user2 AND a.user1 != b.user1 THEN a.user2

END AS common\_friend,

CASE

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 <= b.user2 THEN b.user2

WHEN a.user2 = b.user1 AND a.user1 != b.user2 AND a.user1 > b.user2 THEN a.user1

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 <= b.user1 THEN b.user1

WHEN a.user1 = b.user2 AND a.user2 != b.user1 AND a.user2 > b.user1 THEN a.user2

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 <= b.user2 THEN b.user2

WHEN a.user1 = b.user1 AND a.user2 != b.user2 AND a.user2 > b.user2 THEN a.user2

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 <= b.user1 THEN b.user1

WHEN a.user2 = b.user2 AND a.user1 != b.user1 AND a.user1 > b.user1 THEN a.user1

END AS user2

FROM ord\_tbl a join ord\_tbl b ON

(a.user2 = b.user1 AND a.user1 != b.user2) OR

(a.user1 = b.user2 AND a.user2 != b.user1) OR

(a.user1 = b.user1 AND a.user2 != b.user2) OR

(a.user2 = b.user2 AND a.user1 != b.user1))

SELECT

second\_degree\_tbl.user1 AS user1,

second\_degree\_tbl.common\_friend AS common\_friend,

second\_degree\_tbl.user2 AS user2

FROM second\_degree\_tbl LEFT JOIN ord\_tbl ON (second\_degree\_tbl.user1 = ord\_tbl.user1 AND second\_degree\_tbl.user2 = ord\_tbl.user2)

WHERE ord\_tbl.user1 IS NULL

ORDER BY user1, common\_friend, user2 ;

-- solution 2.

-- trail 1

WITH reverse\_ord\_tbl (user1, user2)

AS (

SELECT user2 AS user1,

user1 AS user2

FROM user\_fb\_HW

),

combine\_tbl (user1, user2)

AS (

SELECT \* from user\_fb\_HW

UNION

SELECT \* from reverse\_ord\_tbl

)

SELECT \* FROM combine\_tbl;

-- trail 2

WITH reverse\_ord\_tbl (user1, user2)

AS (

SELECT user2 AS user1,

user1 AS user2

FROM user\_fb\_HW

),

combine\_tbl (user1, user2)

AS (

SELECT \* from user\_fb\_HW

UNION

SELECT \* from reverse\_ord\_tbl

)

SELECT \*

FROM combine\_tbl t1 JOIN combine\_tbl t2 ON (t1.user2 = t2.user1);

-- trail 3

WITH reverse\_ord\_tbl (user1, user2)

AS (

SELECT user2 AS user1,

user1 AS user2

FROM user\_fb\_HW

),

combine\_tbl (user1, user2)

AS (

SELECT \* from user\_fb\_HW

UNION

SELECT \* from reverse\_ord\_tbl

)

SELECT \*

FROM combine\_tbl t1 JOIN combine\_tbl t2 ON (t1.user2 = t2.user1)

WHERE t1.user1 != t2.user2 AND t1.user1 <= t2.user2;

-- trail 4

WITH reverse\_ord\_tbl (user1, user2)

AS (

SELECT user2 AS user1,

user1 AS user2

FROM user\_fb\_HW

),

combine\_tbl (user1, user2)

AS (

SELECT \* from user\_fb\_HW

UNION

SELECT \* from reverse\_ord\_tbl

),

second\_degree\_tbl (user1, common\_friend, user2)

AS(

SELECT t1.user1, t1.user2, t2.user2

FROM combine\_tbl t1 JOIN combine\_tbl t2 ON (t1.user2 = t2.user1)

WHERE t1.user1 != t2.user2 AND t1.user1 <= t2.user2

)

SELECT

second\_degree\_tbl.user1 AS user1,

second\_degree\_tbl.common\_friend AS common\_friend,

second\_degree\_tbl.user2 AS user2

FROM second\_degree\_tbl LEFT JOIN combine\_tbl ON (second\_degree\_tbl.user1 = combine\_tbl.user1 AND second\_degree\_tbl.user2 = combine\_tbl.user2)

WHERE combine\_tbl.user1 IS NULL

ORDER BY user1, common\_friend, user2 ;