1)Write a solution to find the ids of products that are both low fat and recyclable.

Return the result table in any order.

Input:

Products table:

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

| product\_id | low\_fats | recyclable |

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

| 0 | Y | N |

| 1 | Y | Y |

| 2 | N | Y |

| 3 | Y | Y |

| 4 | N | N |

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

Output:

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

| product\_id |

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

| 1 |

| 3 |

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

ANS :

select product\_id

FROM products

where low\_fats ='Y' and recyclable ='Y'

2)Find the names of the customer that are not referred by the customer with id = 2.

Input:

Customer table:

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

| id | name | referee\_id |

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

| 1 | Will | null |

| 2 | Jane | null |

| 3 | Alex | 2 |

| 4 | Bill | null |

| 5 | Zack | 1 |

| 6 | Mark | 2 |

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

Output:

+------+

| name |

+------+

| Will |

| Jane |

| Bill |

| Zack |

+------+

ANS:

SELECT name

from Customer

where referee\_id IS NULL OR referee\_id <> '2'

3)Write a solution to find the name, population, and area of the big countries.

Input:

World table:

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

| name | continent | area | population | gdp |

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

| Afghanistan | Asia | 652230 | 25500100 | 20343000000 |

| Albania | Europe | 28748 | 2831741 | 12960000000 |

| Algeria | Africa | 2381741 | 37100000 | 188681000000 |

| Andorra | Europe | 468 | 78115 | 3712000000 |

| Angola | Africa | 1246700 | 20609294 | 100990000000 |

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

Output:

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

| name | population | area |

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

| Afghanistan | 25500100 | 652230 |

| Algeria | 37100000 | 2381741 |

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

ANS:

SELECT name, population, area

FROM World

WHERE area >= 3000000 OR population >= 25000000;

4)Write a solution to find all the authors that viewed at least one of their own articles.

Return the result table sorted by id in ascending order.

Input:

Views table:

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

| article\_id | author\_id | viewer\_id | view\_date |

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

| 1 | 3 | 5 | 2019-08-01 |

| 1 | 3 | 6 | 2019-08-02 |

| 2 | 7 | 7 | 2019-08-01 |

| 2 | 7 | 6 | 2019-08-02 |

| 4 | 7 | 1 | 2019-07-22 |

| 3 | 4 | 4 | 2019-07-21 |

| 3 | 4 | 4 | 2019-07-21 |

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

Output:

+------+

| id |

+------+

| 4 |

| 7 |

+------+

ANS:

SELECT DISTINCT author\_id as id

FROM Views

where author\_id = viewer\_id

order by id

5)Write a solution to find the IDs of the invalid tweets. The tweet is invalid if the number of characters used in the content of the tweet is strictly greater than 15.

Input:

Tweets table:

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

| tweet\_id | content |

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

| 1 | Vote for Biden |

| 2 | Let us make America great again! |

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

Output:

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

| tweet\_id |

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

| 2 |

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

ANS:

SELECT tweet\_id

FROM tweets

wheere length(content)> '15'

6)

Write a solution to show the unique ID of each user, If a user does not have a unique ID replace just show null.

Input:

Employees table:

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

| id | name |

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

| 1 | Alice |

| 7 | Bob |

| 11 | Meir |

| 90 | Winston |

| 3 | Jonathan |

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

EmployeeUNI table:

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

| id | unique\_id |

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

| 3 | 1 |

| 11 | 2 |

| 90 | 3 |

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

Output:

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

| unique\_id | name |

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

| null | Alice |

| null | Bob |

| 2 | Meir |

| 3 | Winston |

| 1 | Jonathan |

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

ANS:

SELECT EUNI.unique\_id, E.name

FROM Employees E

LEFT JOIN EmployeeUNI EUNI ON E.id = EUNI.id;

7)Write a solution to report the product\_name, year, and price for each sale\_id in the Sales table.

Input:

Sales table:

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

| sale\_id | product\_id | year | quantity | price |

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

| 1 | 100 | 2008 | 10 | 5000 |

| 2 | 100 | 2009 | 12 | 5000 |

| 7 | 200 | 2011 | 15 | 9000 |

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

Product table:

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

| product\_id | product\_name |

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

| 100 | Nokia |

| 200 | Apple |

| 300 | Samsung |

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

Output:

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

| product\_name | year | price |

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

| Nokia | 2008 | 5000 |

| Nokia | 2009 | 5000 |

| Apple | 2011 | 9000 |

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

ANS :

SELECT p.product\_name, s.year, s.price

FROM product p

JOIN sales s

ON p.product\_id=s.product\_id

8)Write a solution to find all dates' Id with higher temperatures compared to its previous dates (yesterday).

Input:

Weather table:

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

| id | recordDate | temperature |

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

| 1 | 2015-01-01 | 10 |

| 2 | 2015-01-02 | 25 |

| 3 | 2015-01-03 | 20 |

| 4 | 2015-01-04 | 30 |

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

Output:

+----+

| id |

+----+

| 2 |

| 4 |

+----+

ANS :

SELECT w2.id

FROM Weather w1

join Weather w2

ON DATEDIFF (w1.recordDate ,w2.recordDate ) = -1

AND w2.temperature>w1.temperature

9)here is a factory website that has several machines each running the same number of processes. Write a solution to find the average time each machine takes to complete a process.

The time to complete a process is the 'end' timestamp minus the 'start' timestamp. The average time is calculated by the total time to complete every process on the machine divided by the number of processes that were run.

The resulting table should have the machine\_id along with the average time as processing\_time, which should be rounded to 3 decimal places.

Input:

Activity table:

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

| machine\_id | process\_id | activity\_type | timestamp |

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

| 0 | 0 | start | 0.712 |

| 0 | 0 | end | 1.520 |

| 0 | 1 | start | 3.140 |

| 0 | 1 | end | 4.120 |

| 1 | 0 | start | 0.550 |

| 1 | 0 | end | 1.550 |

| 1 | 1 | start | 0.430 |

| 1 | 1 | end | 1.420 |

| 2 | 0 | start | 4.100 |

| 2 | 0 | end | 4.512 |

| 2 | 1 | start | 2.500 |

| 2 | 1 | end | 5.000 |

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

Output:

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

| machine\_id | processing\_time |

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

| 0 | 0.894 |

| 1 | 0.995 |

| 2 | 1.456 |

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

ANS:

SELECT machine\_id,

ROUND(AVG(end\_time - start\_time), 3) AS processing\_time

FROM (

SELECT machine\_id,

process\_id,

MAX(CASE WHEN activity\_type = 'start' THEN timestamp END) AS start\_time,

MAX(CASE WHEN activity\_type = 'end' THEN timestamp END) AS end\_time

FROM Activity

GROUP BY machine\_id, process\_id

) AS process\_times

GROUP BY machine\_id;

10)Write a solution to report the name and bonus amount of each employee with a bonus less than 1000.

Input:

Employee table:

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

| empId | name | supervisor | salary |

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

| 3 | Brad | null | 4000 |

| 1 | John | 3 | 1000 |

| 2 | Dan | 3 | 2000 |

| 4 | Thomas | 3 | 4000 |

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

Bonus table:

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

| empId | bonus |

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

| 2 | 500 |

| 4 | 2000 |

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

Output:

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

| name | bonus |

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

| Brad | null |

| John | null |

| Dan | 500 |

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

ANS :

SELECT e.name,b.bonus

FROM employee e JOIN bonus b

ON e.empId=b.empID

where b.bonus > 1000 or b.bonus IS NULL

11)Write a solution to find the number of times each student attended each exam.

Input:

Students table:

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

| student\_id | student\_name |

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

| 1 | Alice |

| 2 | Bob |

| 13 | John |

| 6 | Alex |

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

Subjects table:

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

| subject\_name |

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

| Math |

| Physics |

| Programming |

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

Examinations table:

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

| student\_id | subject\_name |

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

| 1 | Math |

| 1 | Physics |

| 1 | Programming |

| 2 | Programming |

| 1 | Physics |

| 1 | Math |

| 13 | Math |

| 13 | Programming |

| 13 | Physics |

| 2 | Math |

| 1 | Math |

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

Output:

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

| student\_id | student\_name | subject\_name | attended\_exams |

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

| 1 | Alice | Math | 3 |

| 1 | Alice | Physics | 2 |

| 1 | Alice | Programming | 1 |

| 2 | Bob | Math | 1 |

| 2 | Bob | Physics | 0 |

| 2 | Bob | Programming | 1 |

| 6 | Alex | Math | 0 |

| 6 | Alex | Physics | 0 |

| 6 | Alex | Programming | 0 |

| 13 | John | Math | 1 |

| 13 | John | Physics | 1 |

| 13 | John | Programming | 1 |

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

ANS :

SELECT

s.student\_id,

s.student\_name,

sub.subject\_name,

COUNT(e.subject\_name) AS attended\_exams

FROM Students s

CROSS JOIN Subjects sub

LEFT JOIN Examinations e ON s.student\_id = e.student\_id AND sub.subject\_name = e.subject\_name

GROUP BY s.student\_id, sub.subject\_name

ORDER BY s.student\_id, sub.subject\_name;

12)Write a solution to find managers with at least five direct reports

Input:

Employee table:

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

| id | name | department | managerId |

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

| 101 | John | A | null |

| 102 | Dan | A | 101 |

| 103 | James | A | 101 |

| 104 | Amy | A | 101 |

| 105 | Anne | A | 101 |

| 106 | Ron | B | 101 |

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

Output:

+------+

| name |

+------+

| John |

+------+

ANS :

SELECT name

from employee

where id IN(

SELECT managerId

FROM Employee

GROUP BY managerId

HAVING COUNT(\*) >= 5

)

13)Write a solution to report the movies with an odd-numbered ID and a description that is not "boring".

Input:

Cinema table:

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

| id | movie | description | rating |

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

| 1 | War | great 3D | 8.9 |

| 2 | Science | fiction | 8.5 |

| 3 | irish | boring | 6.2 |

| 4 | Ice song | Fantacy | 8.6 |

| 5 | House card | Interesting | 9.1 |

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

Output:

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

| id | movie | description | rating |

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

| 5 | House card | Interesting | 9.1 |

| 1 | War | great 3D | 8.9 |

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

ANS:

SELECT id, movie, description, rating

FROM Cinema

WHERE id % 2 = 1 AND description != 'boring'

ORDER BY rating DESC;

14)Write a solution to find the average selling price for each product. average\_price should be rounded to 2 decimal places.

Input:

Prices table:

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

| product\_id | start\_date | end\_date | price |

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

| 1 | 2019-02-17 | 2019-02-28 | 5 |

| 1 | 2019-03-01 | 2019-03-22 | 20 |

| 2 | 2019-02-01 | 2019-02-20 | 15 |

| 2 | 2019-02-21 | 2019-03-31 | 30 |

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

UnitsSold table:

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

| product\_id | purchase\_date | units |

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

| 1 | 2019-02-25 | 100 |

| 1 | 2019-03-01 | 15 |

| 2 | 2019-02-10 | 200 |

| 2 | 2019-03-22 | 30 |

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

Output:

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

| product\_id | average\_price |

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

| 1 | 6.96 |

| 2 | 16.96 |

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

ANS:

SELECT p.product\_id, IFNULL(ROUND(SUM(units\*price)/SUM(units),2),0) AS average\_price

FROM Prices p LEFT JOIN UnitsSold u

ON p.product\_id = u.product\_id AND

u.purchase\_date BETWEEN start\_date AND end\_date

group by product\_id

15)Write an SQL query that reports the average experience years of all the employees for each project, rounded to 2 digits

Input:

Project table:

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

| project\_id | employee\_id |

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

| 1 | 1 |

| 1 | 2 |

| 1 | 3 |

| 2 | 1 |

| 2 | 4 |

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

Employee table:

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

| employee\_id | name | experience\_years |

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

| 1 | Khaled | 3 |

| 2 | Ali | 2 |

| 3 | John | 1 |

| 4 | Doe | 2 |

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

Output:

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

| project\_id | average\_years |

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

| 1 | 2.00 |

| 2 | 2.50 |

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

ANS:

select project\_id,

ROUND(AVG(experience\_years), 2) AS average\_years

FROM Project p JOIN Employee e

ON p.employee\_id = e.employee\_id

group by project\_id

16)

Write a solution to find the percentage of the users registered in each contest rounded to two decimals.

Return the result table ordered by percentage in descending order. In case of a tie, order it by contest\_id in ascending order.

Input:

Users table:

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

| user\_id | user\_name |

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

| 6 | Alice |

| 2 | Bob |

| 7 | Alex |

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

Register table:

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

| contest\_id | user\_id |

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

| 215 | 6 |

| 209 | 2 |

| 208 | 2 |

| 210 | 6 |

| 208 | 6 |

| 209 | 7 |

| 209 | 6 |

| 215 | 7 |

| 208 | 7 |

| 210 | 2 |

| 207 | 2 |

| 210 | 7 |

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

Output:

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

| contest\_id | percentage |

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

| 208 | 100.0 |

| 209 | 100.0 |

| 210 | 100.0 |

| 215 | 66.67 |

| 207 | 33.33 |

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

ANS :

select

contest\_id,

round(count(distinct user\_id) \* 100 /(select count(user\_id) from Users) ,2) as percentage

from Register

group by contest\_id

order by percentage desc,contest\_id

17)Write a solution to find each query\_name, the quality and poor\_query\_percentage.

Both quality and poor\_query\_percentage should be rounded to 2 decimal places.

Input:

Queries table:

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

| query\_name | result | position | rating |

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

| Dog | Golden Retriever | 1 | 5 |

| Dog | German Shepherd | 2 | 5 |

| Dog | Mule | 200 | 1 |

| Cat | Shirazi | 5 | 2 |

| Cat | Siamese | 3 | 3 |

| Cat | Sphynx | 7 | 4 |

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

Output:

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

| query\_name | quality | poor\_query\_percentage |

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

| Dog | 2.50 | 33.33 |

| Cat | 0.66 | 33.33 |

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

ANS:

SELECT

query\_name,

ROUND(AVG(rating/position), 2) AS quality,

ROUND(SUM(CASE WHEN rating < 3 THEN 1 ELSE 0 END) / COUNT(\*) \* 100, 2) AS poor\_query\_percentage

FROM

Queries

WHERE query\_name is not null

GROUP BY

query\_name;

18)Write an SQL query to find for each month and country, the number of transactions and their total amount, the number of approved transactions and their total amount.

Input:

Transactions table:

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

| id | country | state | amount | trans\_date |

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

| 121 | US | approved | 1000 | 2018-12-18 |

| 122 | US | declined | 2000 | 2018-12-19 |

| 123 | US | approved | 2000 | 2019-01-01 |

| 124 | DE | approved | 2000 | 2019-01-07 |

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

Output:

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

| month | country | trans\_count | approved\_count | trans\_total\_amount | approved\_total\_amount |

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

| 2018-12 | US | 2 | 1 | 3000 | 1000 |

| 2019-01 | US | 1 | 1 | 2000 | 2000 |

| 2019-01 | DE | 1 | 1 | 2000 | 2000 |

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

ANS:

SELECT SUBSTR(trans\_date,1,7) as month, country, count(id) as trans\_count,

SUM(CASE WHEN state = 'approved' then 1 else 0 END) as approved\_count,

SUM(amount) as trans\_total\_amount,

SUM(CASE WHEN state = 'approved' then amount else 0 END) as approved\_total\_amount

FROM Transactions

GROUP BY month, country

19)If the customer's preferred delivery date is the same as the order date, then the order is called immediate; otherwise, it is called scheduled.

The first order of a customer is the order with the earliest order date that the customer made. It is guaranteed that a customer has precisely one first order.

Write a solution to find the percentage of immediate orders in the first orders of all customers, rounded to 2 decimal places.

Input:

Delivery table:

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

| delivery\_id | customer\_id | order\_date | customer\_pref\_delivery\_date |

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

| 1 | 1 | 2019-08-01 | 2019-08-02 |

| 2 | 2 | 2019-08-02 | 2019-08-02 |

| 3 | 1 | 2019-08-11 | 2019-08-12 |

| 4 | 3 | 2019-08-24 | 2019-08-24 |

| 5 | 3 | 2019-08-21 | 2019-08-22 |

| 6 | 2 | 2019-08-11 | 2019-08-13 |

| 7 | 4 | 2019-08-09 | 2019-08-09 |

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

Output:

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

| immediate\_percentage |

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

| 50.00 |

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

ANS:

Select

round(avg(order\_date = customer\_pref\_delivery\_date)\*100, 2) as immediate\_percentage

from Delivery

where (customer\_id, order\_date) in (

Select customer\_id, min(order\_date)

from Delivery

group by customer\_id

);

20)Write a solution to calculate the number of unique subjects each teacher teaches in the university.

Return the result table in any order.

The result format is shown in the following example.

Example 1:

Input:

Teacher table:

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

| teacher\_id | subject\_id | dept\_id |

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

| 1 | 2 | 3 |

| 1 | 2 | 4 |

| 1 | 3 | 3 |

| 2 | 1 | 1 |

| 2 | 2 | 1 |

| 2 | 3 | 1 |

| 2 | 4 | 1 |

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

Output:

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

| teacher\_id | cnt |

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

| 1 | 2 |

| 2 | 4 |

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

ANS :

SELECT teacher\_id,count(DISTINCT subject\_id) AS cnt

from teacher

group by teacher\_id

21)Write a solution to find the daily active user count for a period of 30 days ending 2019-07-27 inclusively. A user was active on someday if they made at least one activity on that day.

Input:

Activity table:

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

| user\_id | session\_id | activity\_date | activity\_type |

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

| 1 | 1 | 2019-07-20 | open\_session |

| 1 | 1 | 2019-07-20 | scroll\_down |

| 1 | 1 | 2019-07-20 | end\_session |

| 2 | 4 | 2019-07-20 | open\_session |

| 2 | 4 | 2019-07-21 | send\_message |

| 2 | 4 | 2019-07-21 | end\_session |

| 3 | 2 | 2019-07-21 | open\_session |

| 3 | 2 | 2019-07-21 | send\_message |

| 3 | 2 | 2019-07-21 | end\_session |

| 4 | 3 | 2019-06-25 | open\_session |

| 4 | 3 | 2019-06-25 | end\_session |

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

Output:

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

| day | active\_users |

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

| 2019-07-20 | 2 |

| 2019-07-21 | 2 |

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

ANS :

SELECT activity\_date AS day,

COUNT(DISTINCT user\_id) AS active\_users

FROM Activity

WHERE activity\_date BETWEEN '2019-06-28' AND '2019-07-27'

GROUP BY activity\_date;

22)Write a solution to find all the classes that have at least five students.

Input:

Courses table:

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

| student | class |

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

| A | Math |

| B | English |

| C | Math |

| D | Biology |

| E | Math |

| F | Computer |

| G | Math |

| H | Math |

| I | Math |

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

Output:

+---------+

| class |

+---------+

| Math |

+---------+

ANS:

SELECT class

from Courses

group by class

having count(DISTINCT student) >= 5

23)Write a solution that will, for each user, return the number of followers.

Return the result table ordered by user\_id in ascending order.

Input:

Followers table:

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

| user\_id | follower\_id |

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

| 0 | 1 |

| 1 | 0 |

| 2 | 0 |

| 2 | 1 |

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

Output:

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

| user\_id | followers\_count|

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

| 0 | 1 |

| 1 | 1 |

| 2 | 2 |

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

ANS :

SELECT user\_id,

COUNT(follower\_id) AS followers\_count

FROM Followers

GROUP BY user\_id

ORDER BY user\_id;

24)A single number is a number that appeared only once in the MyNumbers table.

Find the largest single number. If there is no single number, report null.

The result format is in the following example.

Input:

MyNumbers table:

+-----+

| num |

+-----+

| 8 |

| 8 |

| 3 |

| 3 |

| 1 |

| 4 |

| 5 |

| 6 |

+-----+

Output:

+-----+

| num |

+-----+

| 6 |

+-----+

Input:

MyNumbers table:

+-----+

| num |

+-----+

| 8 |

| 8 |

| 7 |

| 7 |

| 3 |

| 3 |

| 3 |

+-----+

Output:

+------+

| num |

+------+

| null |

+------+

ANS :

WITH CTE AS (

SELECT num, COUNT(\*) AS count\_num

FROM MyNumbers

GROUP BY num

HAVING COUNT(\*) = 1

)

SELECT max(num) as num

FROM CTE

25)Report for every three line segments whether they can form a triangle.

Input:

Triangle table:

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

| x | y | z |

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

| 13 | 15 | 30 |

| 10 | 20 | 15 |

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

Output:

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

| x | y | z | triangle |

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

| 13 | 15 | 30 | No |

| 10 | 20 | 15 | Yes |

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

ANS :

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