**Select**

1. Table: Customer

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

| Column Name | Type |

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

| id | int |

| name | varchar |

| referee\_id | int |

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

id is the primary key column for this table.

Each row of this table indicates the id of a customer, their name, and the id of the customer who referred them.

Write an SQL query to report the IDs of the customer that are **not referred by** the customer with id = 2.

Return the result table in **any order**.

The query result format is in the following example.

**Example 1:**

**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 |

+------+

Answer:

select name from Customer

where referee\_id !=2 OR referee\_id IS NULL;

referee\_id IS NULL; is very important because MySQL uses three-valued logic -- TRUE, FALSE and UNKNOWN. Anything compared to NULL evaluates to the third value: UNKNOWN. That “anything” includes NULL itself! That’s why MySQL provides the IS NULL and IS NOT NULL operators to specifically check for NULL. Thus, one more condition 'referee\_id IS NULL' should be added to the WHERE clause as below.

Table: Customers

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

| Column Name | Type |

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

| id | int |

| name | varchar |

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

id is the primary key column for this table.

Each row of this table indicates the ID and name of a customer.

Table: Orders

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

| Column Name | Type |

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

| id | int |

| customerId | int |

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

id is the primary key column for this table.

customerId is a foreign key of the ID from the Customers table.

Each row of this table indicates the ID of an order and the ID of the customer who ordered it.

Write an SQL query to report all customers who never order anything.

Return the result table in **any order**.

The query result format is in the following example.

**Example 1:**

**Input:**

Customers table:

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

| id | name |

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

| 1 | Joe |

| 2 | Henry |

| 3 | Sam |

| 4 | Max |

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

Orders table:

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

| id | customerId |

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

| 1 | 3 |

| 2 | 1 |

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

**Output:**

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

| Customers |

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

| Henry |

| Max |

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

Answer:

select Customers.name AS Customers from Customers

Left join Orders on Orders.customerId =Customers.id

WHERE Orders.CustomerId IS NULL

Another solution:

select customers.name as 'Customers'

from customers

where customers.id not in

(

select customerid from orders

);

Inner Join

Only returns connected, matching rows. We might have duplicates in inner join.

Left join

Returns all connected rows, and unconnected rows from the left table. If left table value has no matching with right table value, it brings the left table values in the results and writes null for the right table columns (nulls in right)

Right join

Returns all connected rows, and unconnected rows from right table (nulls in left)

Full join

Left join+right join

Returns all connected rows and unconnected rows from both left and right tables.

Select x from y left/inner/right join y on x.id=y.custid

In above, the table in front of select statement (x) is the table that is located on the left.

**Select & Order**

Calculate Special Bonus

Table: Employees

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

| Column Name | Type |

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

| employee\_id | int |

| name | varchar |

| salary | int |

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

employee\_id is the primary key for this table.

Each row of this table indicates the employee ID, employee name, and salary.

Write an SQL query to calculate the bonus of each employee. The bonus of an employee is 100% of their salary if the ID of the employee is **an odd number** and **the employee name does not start with the character**'M'. The bonus of an employee is 0 otherwise.

Return the result table ordered by employee\_id.

The query result format is in the following example.

**Example 1:**

**Input:**

Employees table:

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

| employee\_id | name | salary |

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

| 2 | Meir | 3000 |

| 3 | Michael | 3800 |

| 7 | Addilyn | 7400 |

| 8 | Juan | 6100 |

| 9 | Kannon | 7700 |

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

**Output:**

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

| employee\_id | bonus |

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

| 2 | 0 |

| 3 | 0 |

| 7 | 7400 |

| 8 | 0 |

| 9 | 7700 |

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

**Explanation:**

The employees with IDs 2 and 8 get 0 bonus because they have an even employee\_id.

The employee with ID 3 gets 0 bonus because their name starts with 'M'.

The rest of the employees get a 100% bonus.

Answer :

select

employee\_id,

case when

employee\_id %2 !=0 and name not like 'M%' then salary

else

0

end

as bonus

from

employees

case when … then comes before from table

Table: Salary

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

| Column Name | Type |

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

| id | int |

| name | varchar |

| sex | ENUM |

| salary | int |

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

id is the primary key for this table.

The sex column is ENUM value of type ('m', 'f').

The table contains information about an employee.

Write an SQL query to 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 temporary tables.

Note that you must write a single update statement, **do not** write any select statement for this problem.

The query result format is in the following example.

**Example 1:**

**Input:**

Salary table:

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

| id | name | sex | salary |

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

| 1 | A | m | 2500 |

| 2 | B | f | 1500 |

| 3 | C | m | 5500 |

| 4 | D | f | 500 |

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

**Output:**

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

| id | name | sex | salary |

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

| 1 | A | f | 2500 |

| 2 | B | m | 1500 |

| 3 | C | f | 5500 |

| 4 | D | m | 500 |

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

**Explanation:**

(1, A) and (3, C) were changed from 'm' to 'f'.

(2, B) and (4, D) were changed from 'f' to 'm'.

Answer: UPDATE salary

SET

sex = CASE sex

WHEN 'm' THEN 'f'

ELSE 'm'

END;

Update and set. We set the sex= case sex when condition then…. Else …