**Question:**

Given a table Products with the following structure:

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

| Column Name | Type |

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

| product\_id | int |

| low\_fats | enum |

| recyclable | enum |

|  |  |
| --- | --- |
|  |  |

* product\_id is the primary key (column with unique values) for this table.
* low\_fats is an ENUM type with values 'Y' or 'N' where 'Y' means the product is low fat and 'N' means it is not.
* recyclable is an ENUM type with values 'Y' or 'N' where 'Y' means the product is recyclable and 'N' means it is not.

Write an SQL query to find the IDs of products that are both low fat and recyclable.

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

| product\_id | low\_fats | recyclable |

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

| 0 | Y | N |

| 1 | Y | Y |

| 2 | N | Y |

| 3 | Y | Y |

| 4 | N | N |

|  |  |  |
| --- | --- | --- |
|  |  |  |

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

| product\_id |

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

| 1 |

| 3 |

**Answer:**

To find the product\_id of products that are both low fat and recyclable, you can use the following SQL query:

SELECT product\_id

FROM Products

WHERE low\_fats = 'Y' AND recyclable = 'Y';

Question

Table: Customer

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

| Column Name | Type |

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

| id | int |

| name | varchar |

| referee\_id | int |

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

In SQL, 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.

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

Return the result table in any order.

The 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 IS NULL OR referee\_id != 2;

Question

Table: World

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

| Column Name | Type |

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

| name | varchar |

| continent | varchar |

| area | int |

| population | int |

| gdp | bigint |

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

name is the primary key (column with unique values) for this table.

Each row of this table gives information about the name of a country, the continent to which it belongs, its area, the population, and its GDP value.

A country is **big** if:

* it has an area of at least three million (i.e., 3000000 km2), or
* it has a population of at least twenty-five million (i.e., 25000000).

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

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

The result format is in the following example.

**Example 1:**

**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 |

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

Answer

select name,population,area from World

where area >=3000000  or population >=25000000;

Question

Table: Views +---------------+---------+ | Column Name | Type | +---------------+---------+ | article\_id | int | | author\_id | int | | viewer\_id | int | | view\_date | date | +---------------+---------+ There is no primary key (column with unique values) for this table, the table may have duplicate rows. Each row of this table indicates that some viewer viewed an article (written by some author) on some date. Note that equal author\_id and viewer\_id indicate the same person. 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. The result format is in the following example. Example 1: 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 | +------+

SELECT DISTINCT author\_id AS id

FROM Views

WHERE author\_id = viewer\_id

ORDER BY author\_id ASC;

Question

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

| Column Name | Type |

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

| tweet\_id | int |

| content | varchar |

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

tweet\_id is the primary key (column with unique values) for this table.

This table contains all the tweets in a social media app.

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.

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

The result format is in the following example.

**Example 1:**

**Input:**

Tweets table:

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

| tweet\_id | content |

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

| 1 | Vote for Biden |

| 2 | Let us make America great again! |

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

**Output:**

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

| tweet\_id |

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

| 2 |

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

**Explanation:**

Tweet 1 has length = 14. It is a valid tweet.

Tweet 2 has length = 32. It is an invalid tweet.

select tweet\_id from Tweets

where length(content)>15;

Question

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

| Column Name | Type |

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

| id | int |

| name | varchar |

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

id is the primary key (column with unique values) for this table.

Each row of this table contains the id and the name of an employee in a company.

Table: EmployeeUNI

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

| Column Name | Type |

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

| id | int |

| unique\_id | int |

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

(id, unique\_id) is the primary key (combination of columns with unique values) for this table.

Each row of this table contains the id and the corresponding unique id of an employee in the company.

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

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

The result format is in the following example.

**Example 1:**

**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 |

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

**Explanation:**

Alice and Bob do not have a unique ID, We will show null instead.

The unique ID of Meir is 2.

The unique ID of Winston is 3.

The unique ID of Jonathan is 1.

select EmployeeUNI.unique\_id,Employees.name from Employees

left join EmployeeUNI

on Employees.id=EmployeeUNI.id;

Question

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

| Column Name | Type |

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

| sale\_id | int |

| product\_id | int |

| year | int |

| quantity | int |

| price | int |

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

(sale\_id, year) is the primary key (combination of columns with unique values) of this table.

product\_id is a foreign key (reference column) to Product table.

Each row of this table shows a sale on the product product\_id in a certain year.

Note that the price is per unit.

Table: Product

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

| Column Name | Type |

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

| product\_id | int |

| product\_name | varchar |

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

product\_id is the primary key (column with unique values) of this table.

Each row of this table indicates the product name of each product.

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

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

The result format is in the following example.

**Example 1:**

**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 |

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

**Explanation:**

From sale\_id = 1, we can conclude that Nokia was sold for 5000 in the year 2008.

From sale\_id = 2, we can conclude that Nokia was sold for 5000 in the year 2009.

From sale\_id = 7, we can conclude that Apple was sold for 9000 in the year 2011.

ANswer

select p.product\_name,s.year,s.price from Sales s

left join Product p

on s.product\_id=p.product\_id;