28/07/2024

1 : Table: Products

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

| 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 (category) of type ('Y', 'N') where 'Y' means this product is low fat and 'N' means it is not.

recyclable is an ENUM (category) of types ('Y', 'N') where 'Y' means this product is recyclable and 'N' means it is not.

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

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

SOLUTION :

SELECT product\_id

FROM Products

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

2 : 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**

SOLUTION:

SELECT name

FROM Customer

WHERE referee\_id != 2 or referee\_id is null ;

3 : 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**.

SOLUTION : select name, population, area

from World

where

area >= 3000000 or

population >= 25000000

4 : 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.

SOLUTION : SELECT DISTINCT author\_id AS id

FROM Views

WHERE author\_id = viewer\_id

ORDER BY id ;

5 : Table: Tweets

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

| 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**.

SOLUTION : SELECT tweet\_id

FROM Tweets

WHERE LENGTH(content) > 15;

6 : Table: Employees

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

| 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.

SOLUTION : SELECT  e.name, eu.unique\_id

FROM Employees e

LEFT JOIN EmployeeUNI eu ON e.id = eu.id;

7 : Table: Sales

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

| 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**.

SOLUTION :

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

FROM Product p

RIGHT JOIN Sales s ON s.product\_id = p.product\_id;

8 : Table: Visits

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

| Column Name | Type |

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

| visit\_id | int |

| customer\_id | int |

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

visit\_id is the column with unique values for this table.

This table contains information about the customers who visited the mall.

Table: Transactions

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

| Column Name | Type |

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

| transaction\_id | int |

| visit\_id | int |

| amount | int |

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

transaction\_id is column with unique values for this table.

This table contains information about the transactions made during the visit\_id.

Write a solution to find the IDs of the users who visited without making any transactions and the number of times they made these types of visits.

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

SOLUTION :

SELECT v.customer\_id, COUNT(v.visit\_id) AS count\_no\_trans

from Visits v

LEFT JOIN Transactions t

ON v.visit\_id = t.visit\_id

WHERE t.transaction\_id IS NULL

GROUP BY v.customer\_id;

8 : Table: Weather

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

| Column Name | Type |

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

| id | int |

| recordDate | date |

| temperature | int |

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

id is the column with unique values for this table.

There are no different rows with the same recordDate.

This table contains information about the temperature on a certain day.

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

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

SOLUTION :

SELECT w1.id

FROM Weather w1

JOIN Weather w2 ON w1.recordDate = DATE\_ADD(w2.recordDate, INTERVAL 1 DAY)

WHERE w1.temperature > w2.temperature;