AMRITA VIDHYA VISHWAPEETHAM UNIVERSITY

DEPARTMENT OF AIE

SEM: V YEAR: III SUBCODE:19CSE202

ASSIGNMENT NO: 01

Question : 01

Table:  World

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

| Column Name | Type |

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

| name | varchar |

| continent | varchar |

| area | int |

| population | int |

| gdp | int |

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

name is the primary key column 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 an SQL query to report the name, population, and area of the **big countries**.

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

The query 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 |

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

Question: 02

Table: Courses

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

| Column Name | Type |

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

| student | varchar |

| class | varchar |

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

(student, class) is the primary key column for this table.

Each row of this table indicates the name of a student and the class in which they are enrolled.

Write an SQL query to report all the classes that have **at least five students**.

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

The query result format is in the following example.

**Example 1:**

**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 |

+---------+

**Explanation:**

- Math has 6 students, so we include it.

- English has 1 student, so we do not include it.

- Biology has 1 student, so we do not include it.

- Computer has 1 student, so we do not include it.

Question : 03

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 names 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 |

+------+

Question : 04

Table: Orders

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

| Column Name | Type |

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

| order\_number | int |

| customer\_number | int |

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

order\_number is the primary key for this table.

This table contains information about the order ID and the customer ID.

Write an SQL query to find the customer\_number for the customer who has placed **the largest number of orders**.

The test cases are generated so that **exactly one customer** will have placed more orders than any other customer.

The query result format is in the following example.

**Example 1:**

**Input:**

Orders table:

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

| order\_number | customer\_number |

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

| 1 | 1 |

| 2 | 2 |

| 3 | 3 |

| 4 | 3 |

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

**Output:**

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

| customer\_number |

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

| 3 |

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

**Explanation:**

The customer with number 3 has two orders, which is greater than either customer 1 or 2 because each of them only has one order.

So the result is customer\_number 3.

Question : 5

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 |

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

**Question: 06**

Table: Employee

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

| Column Name | Type |

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

| id | int |

| name | varchar |

| salary | int |

| departmentId | int |

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

id is the primary key column for this table.

departmentId is a foreign key of the ID from the Department table.

Each row of this table indicates the ID, name, and salary of an employee. It also contains the ID of their department.

Table: Department

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

| Column Name | Type |

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

| id | int |

| name | varchar |

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

id is the primary key column for this table.

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

Write an SQL query to find employees who have the highest salary in each of the departments.

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

The query result format is in the following example.

**Example 1:**

**Input:**

Employee table:

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

| id | name | salary | departmentId |

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

| 1 | Joe | 70000 | 1 |

| 2 | Jim | 90000 | 1 |

| 3 | Henry | 80000 | 2 |

| 4 | Sam | 60000 | 2 |

| 5 | Max | 90000 | 1 |

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

Department table:

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

| id | name |

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

| 1 | IT |

| 2 | Sales |

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

**Output:**

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

| Department | Employee | Salary |

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

| IT | Jim | 90000 |

| Sales | Henry | 80000 |

| IT | Max | 90000 |

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

**Explanation:** Max and Jim both have the highest salary in the IT department and Henry has the highest salary in the Sales department.

Question : 07

Table: Products

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

| Column Name | Type |

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

| product\_id | int |

| low\_fats | enum |

| recyclable | enum |

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

product\_id is the primary key for this table.

low\_fats is an ENUM of type ('Y', 'N') where 'Y' means this product is low fat and 'N' means it is not.

recyclable is an ENUM of types ('Y', 'N') where 'Y' means this 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.

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

The query result format is in the following example.

**Example 1:**

**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 |

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

**Explanation:** Only products 1 and 3 are both low fat and recyclable.

**Question: 08**

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.

**Question : 9**

Table: Person

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

| Column Name | Type |

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

| id | int |

| email | varchar |

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

id is the primary key column for this table.

Each row of this table contains an email. The emails will not contain uppercase letters.

Write an SQL query to report all the duplicate emails.

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

The query result format is in the following example.

**Example 1:**

**Input:**

Person table:

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

| id | email |

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

| 1 | a@b.com |

| 2 | c@d.com |

| 3 | a@b.com |

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

**Output:**

+---------+

| Email |

+---------+

| a@b.com |

+---------+

**Explanation:** a@b.com is repeated two times.

**Question : 10**

Table: SalesPerson

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

| Column Name | Type |

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

| sales\_id | int |

| name | varchar |

| salary | int |

| commission\_rate | int |

| hire\_date | date |

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

sales\_id is the primary key column for this table.

Each row of this table indicates the name and the ID of a salesperson alongside their salary, commission rate, and hire date.

Table: Company

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

| Column Name | Type |

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

| com\_id | int |

| name | varchar |

| city | varchar |

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

com\_id is the primary key column for this table.

Each row of this table indicates the name and the ID of a company and the city in which the company is located.

Table: Orders

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

| Column Name | Type |

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

| order\_id | int |

| order\_date | date |

| com\_id | int |

| sales\_id | int |

| amount | int |

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

order\_id is the primary key column for this table.

com\_id is a foreign key to com\_id from the Company table.

sales\_id is a foreign key to sales\_id from the SalesPerson table.

Each row of this table contains information about one order. This includes the ID of the company, the ID of the salesperson, the date of the order, and the amount paid.

Write an SQL query to report the names of all the salespersons who did not have any orders related to the company with the name **"RED"**.

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

The query result format is in the following example.

**Example 1:**

**Input:**

SalesPerson table:

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

| sales\_id | name | salary | commission\_rate | hire\_date |

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

| 1 | John | 100000 | 6 | 4/1/2006 |

| 2 | Amy | 12000 | 5 | 5/1/2010 |

| 3 | Mark | 65000 | 12 | 12/25/2008 |

| 4 | Pam | 25000 | 25 | 1/1/2005 |

| 5 | Alex | 5000 | 10 | 2/3/2007 |

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

Company table:

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

| com\_id | name | city |

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

| 1 | RED | Boston |

| 2 | ORANGE | New York |

| 3 | YELLOW | Boston |

| 4 | GREEN | Austin |

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

Orders table:

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

| order\_id | order\_date | com\_id | sales\_id | amount |

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

| 1 | 1/1/2014 | 3 | 4 | 10000 |

| 2 | 2/1/2014 | 4 | 5 | 5000 |

| 3 | 3/1/2014 | 1 | 1 | 50000 |

| 4 | 4/1/2014 | 1 | 4 | 25000 |

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

**Output:**

+------+

| name |

+------+

| Amy |

| Mark |

| Alex |

+------+

**Explanation:**

According to orders 3 and 4 in the Orders table, it is easy to tell that only salesperson John and Pam have sales to company RED, so we report all the other names in the table salesperson.

**Question : 11**

Table: Cinema

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

| Column Name | Type |

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

| id | int |

| movie | varchar |

| description | varchar |

| rating | float |

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

id is the primary key for this table.

Each row contains information about the name of a movie, its genre, and its rating.

rating is a 2 decimal places float in the range [0, 10]

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

Return the result table ordered by rating **in descending order**.

The query result format is in the following example.

**Example 1:**

**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 |

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

**Explanation:**

We have three movies with odd-numbered IDs: 1, 3, and 5. The movie with ID = 3 is boring so we do not include it in the answer.

**Hint : Use mod() function**