595. Big Countries

https://leetcode.com/problems/big-countries/description/

Description

```
+-----------------+------------+------------+--------------+---------------+
| name            | continent  | area       | population   | gdp           |
+-----------------+------------+------------+--------------+---------------+
| Afghanistan     | Asia       | 652230     | 25500100     | 20343000      |
| Albania         | Europe     | 28748      | 2831741      | 12960000      |
| Algeria         | Africa     | 2381741    | 37100000     | 188681000     |
| Andorra         | Europe     | 468        | 78115        | 3712000       |
| Angola          | Africa     | 1246700    | 20609294     | 100990000     |
+-----------------+------------+------------+--------------+---------------+
```

表名为wrold 查找面积超过 3,000,000 或者人口数超过 25,000,000 的国家。

输出:

```
+--------------+-------------+--------------+
| name         | population  | area         |
+--------------+-------------+--------------+
| Afghanistan  | 25500100    | 652230       |
| Algeria      | 37100000    | 2381741      |
+--------------+-------------+--------------+
```

Solution

```
SELECT name,
    population,
    area
FROM
    World
WHERE
    area > 3000000
    OR population > 25000000;
```

SQL Schema

SQL Schema 用于在本地环境下创建表结构并导入数据，从而方便在本地环境调试。

```
DROP TABLE
IF
    EXISTS World;
CREATE TABLE World ( NAME VARCHAR ( 255 ), continent VARCHAR ( 255 ), area INT,
population INT, gdp INT );
INSERT INTO World ( NAME, continent, area, population, gdp )
VALUES
    ( 'Afghanistan', 'Asia', '652230', '25500100', '203430000' ),
    ( 'Albania', 'Europe', '28748', '2831741', '129600000' ),
    ( 'Algeria', 'Africa', '2381741', '37100000', '1886810000' ),
    ( 'Andorra', 'Europe', '468', '78115', '37120000' ),
    ( 'Angola', 'Africa', '1246700', '20609294', '1009900000' );
```

627. Swap Salary

https://leetcode.com/problems/swap-salary/description/

Description

```
| id | name | sex | salary |
|----|------|-----|--------|
| 1  | A    | m   | 2500   |
| 2  | B    | f   | 1500   |
| 3  | C    | m   | 5500   |
| 4  | D    | f   | 500    |
```

只用一个 SQL 查询，将 sex 字段反转。

```
| id | name | sex | salary |
|----|------|-----|--------|
| 1  | A    | f   | 2500   |
| 2  | B    | m   | 1500   |
| 3  | C    | f   | 5500   |
| 4  | D    | m   | 500    |
```

Solution

两个相等的数异或的结果为 0，而 0 与任何一个数异或的结果为这个数。

sex 字段只有两个取值：'f' 和 'm'，并且有以下规律：

```
'f' ^ ('m' ^ 'f') = 'm' ^ ('f' ^ 'f') = 'm'
'm' ^ ('m' ^ 'f') = 'f' ^ ('m' ^ 'm') = 'f'
```

因此将 sex 字段和 'm' ^ 'f' 进行异或操作，最后就能反转 sex 字段。

```
UPDATE salary
SET sex = CHAR ( ASCII(sex) ^ ASCII( 'm' ) ^ ASCII( 'f' ) );
```

SQL Schema

```
DROP TABLE
IF
    EXISTS salary;
CREATE TABLE salary ( id INT, NAME VARCHAR ( 100 ), sex CHAR ( 1 ), salary INT
);
INSERT INTO salary ( id, NAME, sex, salary )
VALUES
    ( '1', 'A', 'm', '2500' ),
    ( '2', 'B', 'f', '1500' ),
    ( '3', 'C', 'm', '5500' ),
    ( '4', 'D', 'f', '500' );
```

620. Not Boring Movies

https://leetcode.com/problems/not-boring-movies/description/

Description

```
+---------+-----------+--------------+-----------+
|   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     |
+---------+-----------+--------------+-----------+
```

查找 id 为奇数，并且 description 不是 boring 的电影，按 rating 降序。

```
+---------+-----------+--------------+-----------+
|   id    | movie     |  description |  rating   |
+---------+-----------+--------------+-----------+
|   5     | House card|   Interesting|   9.1     |
|   1     | War       |   great 3D   |   8.9     |
+---------+-----------+--------------+-----------+
```

Solution

```
SELECT
    *
FROM
    cinema
WHERE
    id % 2 = 1
    AND description != 'boring'
ORDER BY
    rating DESC;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS cinema;
CREATE TABLE cinema ( id INT, movie VARCHAR ( 255 ), description VARCHAR ( 255
), rating FLOAT ( 2, 1 ) );
INSERT INTO cinema ( id, movie, description, rating )
VALUES
    ( 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 );
```

596. Classes More Than 5 Students

https://leetcode.com/problems/classes-more-than-5-students/description/

Description

```
+---------+------------+
| student | class      |
+---------+------------+
| A       | Math       |
| B       | English    |
| C       | Math       |
| D       | Biology    |
| E       | Math       |
| F       | Computer   |
| G       | Math       |
| H       | Math       |
| I       | Math       |
+---------+------------+
```

查找有五名及以上 student 的 class。

```
+---------+
| class   |
+---------+
| Math    |
+---------+
```

Solution

对 class 列进行分组之后，再使用 count 汇总函数统计每个分组的记录个数，之后使用 HAVING 进行筛选。HAVING 针对分组进行筛选，而 WHERE 针对每个记录（行）进行筛选。

```sql
SELECT
    class
FROM
    courses
GROUP BY
    class
HAVING
    count( DISTINCT student ) >= 5;
```

SQL Schema

```sql
DROP TABLE
IF
    EXISTS courses;
CREATE TABLE courses ( student VARCHAR ( 255 ), class VARCHAR ( 255 ) );
INSERT INTO courses ( student, class )
VALUES
    ( 'A', 'Math' ),
    ( 'B', 'English' ),
    ( 'C', 'Math' ),
    ( 'D', 'Biology' ),
    ( 'E', 'Math' ),
    ( 'F', 'Computer' ),
    ( 'G', 'Math' ),
    ( 'H', 'Math' ),
    ( 'I', 'Math' );
```

182. Duplicate Emails

https://leetcode.com/problems/duplicate-emails/description/

Description

邮件地址表：

```
+----+---------+
| Id | Email   |
+----+---------+
| 1  | a@b.com |
| 2  | c@d.com |
| 3  | a@b.com |
+----+---------+
```

查找重复的邮件地址：

```
+---------+
| Email   |
+---------+
| a@b.com |
+---------+
```

Solution

对 Email 进行分组，如果并使用 COUNT 进行计数统计，结果大于等于 2 的表示 Email 重复。

```
SELECT
    Email
FROM
    Person
GROUP BY
    Email
HAVING
    COUNT( * ) >= 2;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS Person;
CREATE TABLE Person ( Id INT, Email VARCHAR ( 255 ) );
INSERT INTO Person ( Id, Email )
VALUES
    ( 1, 'a@b.com' ),
    ( 2, 'c@d.com' ),
    ( 3, 'a@b.com' );
```

196. Delete Duplicate Emails

https://leetcode.com/problems/delete-duplicate-emails/description/

Description

邮件地址表：

```
+----+---------+
| Id | Email   |
+----+---------+
| 1  | john@example.com |
| 2  | bob@example.com |
| 3  | john@example.com |
+----+---------+
```

删除重复的邮件地址：

```
+----+------------------+
| Id | Email            |
+----+------------------+
| 1  | john@example.com |
| 2  | bob@example.com  |
+----+------------------+
```

Solution

只保留相同 Email 中 Id 最小的那一个，然后删除其它的。

连接查询：

```
DELETE p1
FROM
    Person p1,
    Person p2
WHERE
    p1.Email = p2.Email
    AND p1.Id > p2.Id
```

子查询：

```
DELETE
FROM
    Person
WHERE
    id NOT IN (
        SELECT id
        FROM (
            SELECT min( id ) AS id
            FROM Person
            GROUP BY email
        ) AS m
    );
```

应该注意的是上述解法额外嵌套了一个 SELECT 语句，如果不这么做，会出现错误：You can't specify target table 'Person' for update in FROM clause。以下演示了这种错误解法。

```
DELETE
FROM
    Person
WHERE
    id NOT IN (
        SELECT min( id ) AS id
        FROM Person
        GROUP BY email
    );
```

参考：[pMySQL Error 1093 - Can't specify target table for update in FROM clause]

SQL Schema

与 182 相同。

175. Combine Two Tables

https://leetcode.com/problems/combine-two-tables/description/

Description

Person 表：

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| PersonId    | int     |
| FirstName   | varchar |
| LastName    | varchar |
+-------------+---------+
PersonId is the primary key column for this table.
```

Address 表：

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| AddressId   | int     |
| PersonId    | int     |
| City        | varchar |
| State       | varchar |
+-------------+---------+
AddressId is the primary key column for this table.
```

查找 FirstName, LastName, City, State 数据，而不管一个用户有没有填地址信息。

Solution

涉及到 Person 和 Address 两个表，在对这两个表执行连接操作时，因为要保留 Person 表中的信息，即使在 Address 表中没有关联的信息也要保留。此时可以用左外连接，将 Person 表放在 LEFT JOIN 的左边。

```
SELECT
    FirstName,
    LastName,
    City,
    State
FROM
    Person P
    LEFT JOIN Address A
    ON P.PersonId = A.PersonId;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS Person;
CREATE TABLE Person ( PersonId INT, FirstName VARCHAR ( 255 ), LastName VARCHAR
( 255 ) );
DROP TABLE
IF
    EXISTS Address;
CREATE TABLE Address ( AddressId INT, PersonId INT, City VARCHAR ( 255 ), State
VARCHAR ( 255 ) );
INSERT INTO Person ( PersonId, LastName, FirstName )
VALUES
    ( 1, 'Wang', 'Allen' );
```

```
INSERT INTO Address ( AddressId, PersonId, City, State )
VALUES
    ( 1, 2, 'New York City', 'New York' );
```

181. Employees Earning More Than Their Managers

https://leetcode.com/problems/employees-earning-more-than-their-managers/description/

Description

Employee 表：

```
+----+-------+--------+-----------+
| Id | Name  | Salary | ManagerId |
+----+-------+--------+-----------+
| 1  | Joe   | 70000  | 3         |
| 2  | Henry | 80000  | 4         |
| 3  | Sam   | 60000  | NULL      |
| 4  | Max   | 90000  | NULL      |
+----+-------+--------+-----------+
```

查找薪资大于其经理薪资的员工信息。

Solution

```
SELECT
    E1.NAME AS Employee
FROM
    Employee E1
    INNER JOIN Employee E2
    ON E1.ManagerId = E2.Id
    AND E1.Salary > E2.Salary;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS Employee;
CREATE TABLE Employee ( Id INT, NAME VARCHAR ( 255 ), Salary INT, ManagerId INT
);
INSERT INTO Employee ( Id, NAME, Salary, ManagerId )
VALUES
    ( 1, 'Joe', 70000, 3 ),
    ( 2, 'Henry', 80000, 4 ),
    ( 3, 'Sam', 60000, NULL ),
    ( 4, 'Max', 90000, NULL );
```

183. Customers Who Never Order

https://leetcode.com/problems/customers-who-never-order/description/

Description

Customers 表：

```
+----+-------+
| Id | Name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
+----+-------+
```

Orders 表：

```
+----+------------+
| Id | CustomerId |
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

查找没有订单的顾客信息：

```
+-----------+
| Customers |
+-----------+
| Henry     |
| Max       |
+-----------+
```

Solution

左外链接

```sql
SELECT
    C.Name AS Customers
FROM
    Customers C
    LEFT JOIN Orders O
    ON C.Id = O.CustomerId
WHERE
    O.CustomerId IS NULL;
```

子查询

```sql
SELECT
    Name AS Customers
FROM
    Customers
WHERE
    Id NOT IN (
        SELECT CustomerId
        FROM Orders
    );
```

SQL Schema

```sql
DROP TABLE
IF
    EXISTS Customers;
CREATE TABLE Customers ( Id INT, NAME VARCHAR ( 255 ) );
DROP TABLE
IF
    EXISTS Orders;
CREATE TABLE Orders ( Id INT, CustomerId INT );
INSERT INTO Customers ( Id, NAME )
VALUES
    ( 1, 'Joe' ),
    ( 2, 'Henry' ),
    ( 3, 'Sam' ),
    ( 4, 'Max' );
INSERT INTO Orders ( Id, CustomerId )
VALUES
    ( 1, 3 ),
    ( 2, 1 );
```

184. Department Highest Salary

https://leetcode.com/problems/department-highest-salary/description/

Description

Employee 表：

```
+----+-------+--------+--------------+
| Id | Name  | Salary | DepartmentId |
+----+-------+--------+--------------+
| 1  | Joe   | 70000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
+----+-------+--------+--------------+
```

Department 表：

```
+----+----------+
| Id | Name     |
+----+----------+
| 1  | IT       |
| 2  | Sales    |
+----+----------+
```

查找一个 Department 中收入最高者的信息：

```
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| Sales      | Henry    | 80000  |
+------------+----------+--------+
```

Solution

创建一个临时表，包含了部门员工的最大薪资。可以对部门进行分组，然后使用 MAX() 汇总函数取得最大薪资。

之后使用连接找到一个部门中薪资等于临时表中最大薪资的员工。

```
SELECT
    D.NAME Department,
    E.NAME Employee,
    E.Salary
FROM
    Employee E,
    Department D,
    ( SELECT DepartmentId, MAX( Salary ) Salary
     FROM Employee
     GROUP BY DepartmentId ) M
WHERE
    E.DepartmentId = D.Id
    AND E.DepartmentId = M.DepartmentId
    AND E.Salary = M.Salary;
```

SQL Schema

```
DROP TABLE IF EXISTS Employee;
CREATE TABLE Employee ( Id INT, NAME VARCHAR ( 255 ), Salary INT, DepartmentId
INT );
DROP TABLE IF EXISTS Department;
CREATE TABLE Department ( Id INT, NAME VARCHAR ( 255 ) );
INSERT INTO Employee ( Id, NAME, Salary, DepartmentId )
VALUES
    ( 1, 'Joe', 70000, 1 ),
    ( 2, 'Henry', 80000, 2 ),
    ( 3, 'Sam', 60000, 2 ),
    ( 4, 'Max', 90000, 1 );
INSERT INTO Department ( Id, NAME )
VALUES
    ( 1, 'IT' ),
    ( 2, 'Sales' );
```

176. Second Highest Salary

https://leetcode.com/problems/second-highest-salary/description/

Description

```
+----+--------+
| Id | Salary |
+----+--------+
| 1  | 100    |
| 2  | 200    |
| 3  | 300    |
+----+--------+
```

查找工资第二高的员工。

```
+---------------------+
| SecondHighestSalary |
+---------------------+
| 200                 |
+---------------------+
```

没有找到返回 null 而不是不返回数据。

Solution

为了在没有查找到数据时返回 null，需要在查询结果外面再套一层 SELECT。

```
SELECT
    ( SELECT DISTINCT Salary
     FROM Employee
     ORDER BY Salary DESC
     LIMIT 1, 1 ) SecondHighestSalary;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS Employee;
CREATE TABLE Employee ( Id INT, Salary INT );
INSERT INTO Employee ( Id, Salary )
VALUES
    ( 1, 100 ),
    ( 2, 200 ),
    ( 3, 300 );
```

177. Nth Highest Salary

Description

查找工资第 N 高的员工。

Solution

```
CREATE FUNCTION getNthHighestSalary ( N INT ) RETURNS INT BEGIN

SET N = N - 1;
RETURN (
    SELECT (
        SELECT DISTINCT Salary
        FROM Employee
        ORDER BY Salary DESC
        LIMIT N, 1
    )
);

END
```

SQL Schema

同 176。

178. Rank Scores

https://leetcode.com/problems/rank-scores/description/

Description

得分表：

```
+----+-------+
| Id | Score |
+----+-------+
| 1  | 3.50  |
| 2  | 3.65  |
| 3  | 4.00  |
| 4  | 3.85  |
| 5  | 4.00  |
| 6  | 3.65  |
+----+-------+
```

将得分排序，并统计排名。

```
+-------+------+
| Score | Rank |
+-------+------+
| 4.00  | 1    |
| 4.00  | 1    |
| 3.85  | 2    |
| 3.65  | 3    |
| 3.65  | 3    |
| 3.50  | 4    |
+-------+------+
```

Solution

要统计某个 score 的排名，只要统计大于等于该 score 的 score 数量。

Id	score	大于等于该 score 的 score 数量	排名
1	4.1	3	3
2	4.2	2	2
3	4.3	1	1

使用连接操作找到某个 score 对应的大于等于其值的记录：

```
SELECT
    *
FROM
    Scores S1
    INNER JOIN Scores S2
    ON S1.score <= S2.score
ORDER BY
    S1.score DESC, S1.Id;
```

S1.Id	S1.score	S2.Id	S2.score
3	4.3	3	4.3
2	4.2	2	4.2
2	4.2	3	4.3
1	4.1	1	4.1
1	4.1	2	4.2
1	4.1	3	4.3

可以看到每个 S1.score 都有对应好几条记录，我们再进行分组，并统计每个分组的数量作为 'Rank'

```
SELECT
    S1.score 'Score',
    COUNT(*) 'Rank'
FROM
    Scores S1
    INNER JOIN Scores S2
    ON S1.score <= S2.score
GROUP BY
    S1.id, S1.score
ORDER BY
    S1.score DESC, S1.Id;
```

score	Rank
4.3	1
4.2	2
4.1	3

上面的解法看似没问题，但是对于以下数据，它却得到了错误的结果：

Id	score
1	4.1
2	4.2
3	4.2

score	Rank
4.2	2
4.2	2
4.1	3

而我们希望的结果为：

score	Rank
4.2	1
4.2	1
4.1	2

连接情况如下：

S1.Id	S1.score	S2.Id	S2.score
2	4.2	3	4.2
2	4.2	2	4.2
3	4.2	3	4.2
3	4.2	2	4.1
1	4.1	3	4.2
1	4.1	2	4.2
1	4.1	1	4.1

我们想要的结果是，把分数相同的放在同一个排名，并且相同分数只占一个位置，例如上面的分数，Id=2 和 Id=3 的记录都有相同的分数，并且最高，他们并列第一。而 Id=1 的记录应该排第二名，而不是第三名。所以在进行 COUNT 计数统计时，我们需要使用 COUNT(DISTINCT S2.score) 从而只统计一次相同的分数。

```
SELECT
    S1.score 'Score',
    COUNT( DISTINCT S2.score ) 'Rank'
FROM
    Scores S1
    INNER JOIN Scores S2
    ON S1.score <= S2.score
GROUP BY
    S1.id, S1.score
ORDER BY
    S1.score DESC;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS Scores;
CREATE TABLE Scores ( Id INT, Score DECIMAL ( 3, 2 ) );
INSERT INTO Scores ( Id, Score )
VALUES
    ( 1, 4.1 ),
    ( 2, 4.1 ),
    ( 3, 4.2 ),
    ( 4, 4.2 ),
    ( 5, 4.3 ),
    ( 6, 4.3 );
```

180. Consecutive Numbers

https://leetcode.com/problems/consecutive-numbers/description/

Description

数字表：

```
+----+-----+
| Id | Num |
+----+-----+
| 1  |  1  |
| 2  |  1  |
| 3  |  1  |
| 4  |  2  |
| 5  |  1  |
| 6  |  2  |
| 7  |  2  |
+----+-----+
```

查找连续出现三次的数字。

```
+-----------------+
| ConsecutiveNums |
+-----------------+
| 1               |
+-----------------+
```

Solution

```sql
SELECT
    DISTINCT L1.num ConsecutiveNums
FROM
    Logs L1,
    Logs L2,
    Logs L3
WHERE L1.id = l2.id - 1
    AND L2.id = L3.id - 1
    AND L1.num = L2.num
    AND l2.num = l3.num;
```

SQL Schema

```sql
DROP TABLE
IF
    EXISTS LOGS;
CREATE TABLE LOGS ( Id INT, Num INT );
INSERT INTO LOGS ( Id, Num )
VALUES
    ( 1, 1 ),
    ( 2, 1 ),
    ( 3, 1 ),
    ( 4, 2 ),
    ( 5, 1 ),
    ( 6, 2 ),
    ( 7, 2 );
```

626. Exchange Seats

https://leetcode.com/problems/exchange-seats/description/

Description

seat 表存储着座位对应的学生。

```
+---------+---------+
|    id   | student |
+---------+---------+
|    1    | Abbot   |
|    2    | Doris   |
|    3    | Emerson |
|    4    | Green   |
|    5    | Jeames  |
+---------+---------+
```

要求交换相邻座位的两个学生，如果最后一个座位是奇数，那么不交换这个座位上的学生。

```
+---------+---------+
|    id   | student |
+---------+---------+
|    1    | Doris   |
|    2    | Abbot   |
|    3    | Green   |
|    4    | Emerson |
|    5    | Jeames  |
+---------+---------+
```

Solution

使用多个 union。

```
# 处理偶数 id，让 id 减 1
# 例如 2,4,6,... 变成 1,3,5,...
SELECT
    s1.id - 1 AS id,
    s1.student
FROM
    seat s1
WHERE
    s1.id MOD 2 = 0 UNION
# 处理奇数 id，让 id 加 1。但是如果最大的 id 为奇数，则不做处理
# 例如 1,3,5,... 变成 2,4,6,...
SELECT
    s2.id + 1 AS id,
    s2.student
FROM
    seat s2
WHERE
    s2.id MOD 2 = 1
    AND s2.id != ( SELECT max( s3.id ) FROM seat s3 ) UNION
# 如果最大的 id 为奇数，单独取出这个数
SELECT
    s4.id AS id,
    s4.student
FROM
    seat s4
WHERE
    s4.id MOD 2 = 1
    AND s4.id = ( SELECT max( s5.id ) FROM seat s5 )
ORDER BY
    id;
```

SQL Schema

```
DROP TABLE
IF
    EXISTS seat;
CREATE TABLE seat ( id INT, student VARCHAR ( 255 ) );
INSERT INTO seat ( id, student )
VALUES
    ( '1', 'Abbot' ),
    ( '2', 'Doris' ),
    ( '3', 'Emerson' ),
    ( '4', 'Green' ),
    ( '5', 'Jeames' );
```