

Leetcode-Database 题解

原作者github: <https://github.com/CyC2018/CS-Notes>

PDF制作github: <https://github.com/sjsdfg/CS-Notes-PDF>

595. Big Countries

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

Description

```
1.  +-----+-----+-----+-----+
2.  | name          | continent | area      | population | gdp
3.  +-----+-----+-----+-----+
4.  | Afghanistan  | Asia      | 652230     | 25500100    | 20343000
5.  | Albania       | Europe    | 28748       | 2831741     | 12960000
6.  | Algeria       | Africa    | 2381741     | 37100000    | 188681000
7.  | Andorra        | Europe    | 468         | 78115        | 3712000
8.  | Angola         | Africa    | 1246700     | 20609294    | 100990000
9.  +-----+-----+-----+-----+
    |
```

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

```
1.  +-----+-----+-----+
2.  | name          | population | area      |
3.  +-----+-----+-----+
4.  | Afghanistan  | 25500100   | 652230    |
```

```

5. | Algeria      | 37100000 | 2381741 |
6. +-----+-----+-----+

```

SQL Schema

```

1. DROP TABLE
2. IF
3.     EXISTS World;
4. CREATE TABLE World ( NAME VARCHAR ( 255 ), continent VARCHAR ( 255 ), a
   rea INT, population INT, gdp INT );
5. INSERT INTO World ( NAME, continent, area, population, gdp )
6. VALUES
7.     ( 'Afghanistan', 'Asia', '652230', '25500100', '203430000' ),
8.     ( 'Albania', 'Europe', '28748', '2831741', '129600000' ),
9.     ( 'Algeria', 'Africa', '2381741', '37100000', '1886810000' ),
10.    ( 'Andorra', 'Europe', '468', '78115', '37120000' ),
11.    ( 'Angola', 'Africa', '1246700', '20609294', '1009900000' );

```

Solution

```

1. SELECT name,
2.     population,
3.     area
4. FROM
5.     World
6. WHERE
7.     area > 3000000
8.     OR population > 25000000;

```

627. Swap Salary

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

Description

```

1. | id | name | sex | salary |
2. |----|-----|-----|-----|
3. | 1  | A    | m   | 2500   |
4. | 2  | B    | f   | 1500   |
5. | 3  | C    | m   | 5500   |
6. | 4  | D    | f   | 500    |

```

只用一个 SQL 查询，将 sex 字段反转。

```

1. | id | name | sex | salary |
2. |----|-----|-----|-----|
3. | 1  | A    | f   | 2500   |
4. | 2  | B    | m   | 1500   |
5. | 3  | C    | f   | 5500   |
6. | 4  | D    | m   | 500    |

```

SQL Schema

```

1. DROP TABLE
2. IF
3.     EXISTS salary;
4. CREATE TABLE salary ( id INT, NAME VARCHAR ( 100 ), sex CHAR ( 1 ), sal
   ary INT );
5. INSERT INTO salary ( id, NAME, sex, salary )
6. VALUES
7.     ( '1', 'A', 'm', '2500' ),
8.     ( '2', 'B', 'f', '1500' ),
9.     ( '3', 'C', 'm', '5500' ),
10.    ( '4', 'D', 'f', '500' );

```

Solution

```

1. UPDATE salary
2. SET sex = CHAR ( ASCII(sex) ^ ASCII( 'm' ) ^ ASCII( 'f' ) );

```

620. Not Boring Movies

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

Description

```
1.  +-----+-----+-----+-----+
2.  |   id   | movie   | description | rating  |
3.  +-----+-----+-----+-----+
4.  |    1    | War     | great 3D   | 8.9     |
5.  |    2    | Science | fiction     | 8.5     |
6.  |    3    | irish   | boring     | 6.2     |
7.  |    4    | Ice song| Fantasy    | 8.6     |
8.  |    5    | House card| Interesting| 9.1     |
9.  +-----+-----+-----+-----+
```

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

```
1.  +-----+-----+-----+-----+
2.  |   id   | movie   | description | rating  |
3.  +-----+-----+-----+-----+
4.  |    5    | House card| Interesting| 9.1     |
5.  |    1    | War     | great 3D   | 8.9     |
6.  +-----+-----+-----+-----+
```

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS cinema;
4.  CREATE TABLE cinema ( id INT, movie VARCHAR ( 255 ), description
5.      VARCHAR ( 255 ), rating FLOAT ( 2, 1 ) );
6.  INSERT INTO cinema ( id, movie, description, rating )
7.  VALUES
8.      ( 1, 'War', 'great 3D', 8.9 ),
9.      ( 2, 'Science', 'fiction', 8.5 ),
10.     ( 3, 'irish', 'boring', 6.2 ),
11.     ( 4, 'Ice song', 'Fantasy', 8.6 ),
```

```
11.      ( 5, 'House card', 'Interesting', 9.1 );
```

Solution

```
1.  SELECT
2.      *
3.  FROM
4.      cinema
5.  WHERE
6.      id % 2 = 1
7.      AND description != 'boring'
8.  ORDER BY
9.      rating DESC;
```

596. Classes More Than 5 Students

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

Description

```
1.  +-----+-----+
2.  | student | class      |
3.  +-----+-----+
4.  | A       | Math       |
5.  | B       | English    |
6.  | C       | Math       |
7.  | D       | Biology    |
8.  | E       | Math       |
9.  | F       | Computer   |
10. | G       | Math       |
11. | H       | Math       |
12. | I       | Math       |
13. +-----+-----+
```

查找有五名及以上 student 的 class。

```
1.  +-----+
2.  | class  |
3.  +-----+
4.  | Math   |
5.  +-----+
```

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS courses;
4.  CREATE TABLE courses ( student VARCHAR ( 255 ), class VARCHAR ( 255 ) )
5.  ;
6.  INSERT INTO courses ( student, class )
7.  VALUES
8.      ( 'A', 'Math' ),
9.      ( 'B', 'English' ),
10.     ( 'C', 'Math' ),
11.     ( 'D', 'Biology' ),
12.     ( 'E', 'Math' ),
13.     ( 'F', 'Computer' ),
14.     ( 'G', 'Math' ),
15.     ( 'H', 'Math' ),
16.     ( 'I', 'Math' );
```

Solution

```
1.  SELECT
2.      class
3.  FROM
4.      courses
5.  GROUP BY
6.      class
7.  HAVING
8.      count( DISTINCT student ) >= 5;
```

182. Duplicate Emails

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

Description

邮件地址表：

```
1.  +-----+-----+
2.  | Id | Email |
3.  +-----+-----+
4.  | 1  | a@b.com |
5.  | 2  | c@d.com |
6.  | 3  | a@b.com |
7.  +-----+-----+
```

查找重复的邮件地址：

```
1.  +-----+
2.  | Email |
3.  +-----+
4.  | a@b.com |
5.  +-----+
```

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS Person;
4.  CREATE TABLE Person ( Id INT, Email VARCHAR ( 255 ) );
5.  INSERT INTO Person ( Id, Email )
6.  VALUES
7.      ( 1, 'a@b.com' ),
8.      ( 2, 'c@d.com' ),
9.      ( 3, 'a@b.com' );
```

Solution

```
1.  SELECT
2.      Email
3.  FROM
4.      Person
5.  GROUP BY
6.      Email
7.  HAVING
8.      COUNT ( * ) >= 2;
```

196. Delete Duplicate Emails

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

Description

邮件地址表：

```
1.  +----+-----+
2.  | Id | Email   |
3.  +----+-----+
4.  | 1  | a@b.com |
5.  | 2  | c@d.com |
6.  | 3  | a@b.com |
7.  +----+-----+
```

删除重复的邮件地址：

```
1.  +----+-----+
2.  | Id | Email           |
3.  +----+-----+
4.  | 1  | john@example.com |
5.  | 2  | bob@example.com  |
6.  +----+-----+
```


SQL Schema

与 182 相同。

Solution

连接：

```
1. DELETE p1
2. FROM
3.     Person p1,
4.     Person p2
5. WHERE
6.     p1.Email = p2.Email
7.     AND p1.Id > p2.Id
```

子查询：

```
1. DELETE
2. FROM
3.     Person
4. WHERE
5.     id NOT IN ( SELECT id FROM ( SELECT min( id ) AS id FROM Person GRO
UP 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。

```
1. DELETE
2. FROM
3.     Person
4. WHERE
5.     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](#)

175. Combine Two Tables

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

Description

Person 表：

```
1.  +-----+-----+
2.  | Column Name | Type      |
3.  +-----+-----+
4.  | PersonId    | int       |
5.  | FirstName   | varchar   |
6.  | LastName    | varchar   |
7.  +-----+-----+
8.  PersonId is the primary key column for this table.
```

Address 表：

```
1.  +-----+-----+
2.  | Column Name | Type      |
3.  +-----+-----+
4.  | AddressId   | int       |
5.  | PersonId    | int       |
6.  | City        | varchar   |
7.  | State       | varchar   |
8.  +-----+-----+
9.  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。

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS Person;
4.  CREATE TABLE Person ( PersonId INT, FirstName VARCHAR ( 255 ), LastName
```

```

5.      VARCHAR ( 255 ) );
6.      DROP TABLE
7.      IF
8.      EXISTS Address;
9.      CREATE TABLE Address ( AddressId INT, PersonId INT, City VARCHAR ( 255
10.     ), State VARCHAR ( 255 ) );
11.     INSERT INTO Person ( PersonId, LastName, FirstName )
12.     VALUES
13.     ( 1, 'Wang', 'Allen' );
14.     INSERT INTO Address ( AddressId, PersonId, City, State )
15.     VALUES
16.     ( 1, 2, 'New York City', 'New York' );

```

Solution

使用左外连接。

```

1.      SELECT
2.      FirstName,
3.      LastName,
4.      City,
5.      State
6.      FROM
7.      Person P
8.      LEFT JOIN Address A
9.      ON P.PersonId = A.PersonId;

```

181. Employees Earning More Than Their Managers

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

Description

Employee 表：

```

1.  +-----+-----+-----+-----+
2.  | Id | Name  | Salary | ManagerId |
3.  +-----+-----+-----+-----+
4.  | 1  | Joe   | 70000  | 3         |
5.  | 2  | Henry | 80000  | 4         |
6.  | 3  | Sam   | 60000  | NULL      |
7.  | 4  | Max   | 90000  | NULL      |
8.  +-----+-----+-----+-----+

```

查找薪资大于其经理薪资的员工信息。

SQL Schema

```

1.  DROP TABLE
2.  IF
3.      EXISTS Employee;
4.  CREATE TABLE Employee ( Id INT, NAME VARCHAR ( 255 ), Salary INT, Manag
5.  erId INT );
6.  INSERT INTO Employee ( Id, NAME, Salary, ManagerId )
7.  VALUES
8.      ( 1, 'Joe', 70000, 3 ),
9.      ( 2, 'Henry', 80000, 4 ),
10.     ( 3, 'Sam', 60000, NULL ),
11.     ( 4, 'Max', 90000, NULL );

```

Solution

```

1.  SELECT
2.      E1.NAME AS Employee
3.  FROM
4.      Employee E1
5.      INNER JOIN Employee E2
6.      ON E1.ManagerId = E2.Id
7.      AND E1.Salary > E2.Salary;

```

183. Customers Who Never Order

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

Description

Customers 表：

```
1.  +-----+-----+
2.  | Id | Name |
3.  +-----+-----+
4.  | 1 | Joe |
5.  | 2 | Henry |
6.  | 3 | Sam |
7.  | 4 | Max |
8.  +-----+-----+
```

Orders 表：

```
1.  +-----+-----+
2.  | Id | CustomerId |
3.  +-----+-----+
4.  | 1 | 3 |
5.  | 2 | 1 |
6.  +-----+-----+
```

查找没有订单的顾客信息：

```
1.  +-----+
2.  | Customers |
3.  +-----+
4.  | Henry |
5.  | Max |
6.  +-----+
```

SQL Schema

```

1. DROP TABLE
2. IF
3.     EXISTS Customers;
4. CREATE TABLE Customers ( Id INT, NAME VARCHAR ( 255 ) );
5. DROP TABLE
6. IF
7.     EXISTS Orders;
8. CREATE TABLE Orders ( Id INT, CustomerId INT );
9. INSERT INTO Customers ( Id, NAME )
10. VALUES
11.     ( 1, 'Joe' ),
12.     ( 2, 'Henry' ),
13.     ( 3, 'Sam' ),
14.     ( 4, 'Max' );
15. INSERT INTO Orders ( Id, CustomerId )
16. VALUES
17.     ( 1, 3 ),
18.     ( 2, 1 );

```

Solution

左外链接

```

1. SELECT
2.     C.Name AS Customers
3. FROM
4.     Customers C
5.     LEFT JOIN Orders O
6.         ON C.Id = O.CustomerId
7. WHERE
8.     O.CustomerId IS NULL;

```

子查询

```

1. SELECT
2.     Name AS Customers
3. FROM
4.     Customers
5. WHERE
6.     Id NOT IN ( SELECT CustomerId FROM Orders );

```

184. Department Highest Salary

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

Description

Employee 表：

```
1.  +-----+-----+-----+-----+
2.  | Id | Name  | Salary | DepartmentId |
3.  +-----+-----+-----+-----+
4.  | 1  | Joe   | 70000  | 1             |
5.  | 2  | Henry | 80000  | 2             |
6.  | 3  | Sam   | 60000  | 2             |
7.  | 4  | Max   | 90000  | 1             |
8.  +-----+-----+-----+-----+
```

Department 表：

```
1.  +-----+-----+
2.  | Id | Name  |
3.  +-----+-----+
4.  | 1  | IT    |
5.  | 2  | Sales |
6.  +-----+-----+
```

查找一个 Department 中收入最高者的信息：

```
1.  +-----+-----+-----+
2.  | Department | Employee | Salary |
3.  +-----+-----+-----+
4.  | IT         | Max      | 90000  |
5.  | Sales      | Henry    | 80000  |
6.  +-----+-----+-----+
```

SQL Schema

```

1. DROP TABLE IF EXISTS Employee;
2. CREATE TABLE Employee ( Id INT, NAME VARCHAR ( 255 ), Salary INT, DepartmentId INT );
3. DROP TABLE IF EXISTS Department;
4. CREATE TABLE Department ( Id INT, NAME VARCHAR ( 255 ) );
5. INSERT INTO Employee ( Id, NAME, Salary, DepartmentId )
6. VALUES
7.     ( 1, 'Joe', 70000, 1 ),
8.     ( 2, 'Henry', 80000, 2 ),
9.     ( 3, 'Sam', 60000, 2 ),
10.    ( 4, 'Max', 90000, 1 );
11. INSERT INTO Department ( Id, NAME )
12. VALUES
13.     ( 1, 'IT' ),
14.     ( 2, 'Sales' );

```

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。

```

1. SELECT
2.     D.NAME Department,
3.     E.NAME Employee,
4.     E.Salary
5. FROM
6.     Employee E,
7.     Department D,
8.     ( SELECT DepartmentId, MAX( Salary ) Salary FROM Employee GROUP BY
9.         DepartmentId ) M
10. WHERE
11.     E.DepartmentId = D.Id
12.     AND E.DepartmentId = M.DepartmentId
13.     AND E.Salary = M.Salary;

```

176. Second Highest Salary

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

Description

```
1.  +-----+-----+
2.  | Id | Salary |
3.  +-----+-----+
4.  | 1  | 100   |
5.  | 2  | 200   |
6.  | 3  | 300   |
7.  +-----+-----+
```

查找工资第二高的员工。

```
1.  +-----+
2.  | SecondHighestSalary |
3.  +-----+
4.  | 200                  |
5.  +-----+
```

没有找到返回 null 而不是不返回数据。

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS Employee;
4.  CREATE TABLE Employee ( Id INT, Salary INT );
5.  INSERT INTO Employee ( Id, Salary )
6.  VALUES
7.      ( 1, 100 ),
8.      ( 2, 200 ),
9.      ( 3, 300 );
```

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。

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

177. Nth Highest Salary

Description

查找工资第 N 高的员工。

SQL Schema

同 176。

Solution

```
1. CREATE FUNCTION getNthHighestSalary ( N INT ) RETURNS INT BEGIN
2.
3.     SET N = N - 1;
4.     RETURN ( SELECT ( SELECT DISTINCT Salary FROM Employee ORDER BY Salary
                        DESC LIMIT N, 1 ) );
5.
6.     END
```

178. Rank Scores

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

Description

得分表：

1.	+-----+-----+
2.	Id Score
3.	+-----+-----+
4.	1 3.50
5.	2 3.65
6.	3 4.00
7.	4 3.85
8.	5 4.00
9.	6 3.65
10.	+-----+-----+

将得分排序，并统计排名。

1.	+-----+-----+
2.	Score Rank
3.	+-----+-----+
4.	4.00 1
5.	4.00 1
6.	3.85 2
7.	3.65 3
8.	3.65 3
9.	3.50 4
10.	+-----+-----+

SQL Schema

```
1. DROP TABLE
2. IF
3.     EXISTS Scores;
4. CREATE TABLE Scores ( Id INT, Score DECIMAL ( 3, 2 ) );
5. INSERT INTO Scores ( Id, Score )
6. VALUES
7.     ( 1, 3.5 ),
8.     ( 2, 3.65 ),
9.     ( 3, 4.0 ),
```

```
10.      ( 4, 3.85 ),
11.      ( 5, 4.0 ),
12.      ( 6, 3.65 );
```

Solution

```
1.  SELECT
2.      S1.score,
3.      COUNT( DISTINCT S2.score ) Rank
4.  FROM
5.      Scores S1
6.      INNER JOIN Scores S2
7.      ON S1.score <= S2.score
8.  GROUP BY
9.      S1.id
10. ORDER BY
11.     S1.score DESC;
```

180. Consecutive Numbers

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

Description

数字表：

```
1.  +-----+-----+
2.  | Id | Num |
3.  +-----+-----+
4.  | 1 | 1 |
5.  | 2 | 1 |
6.  | 3 | 1 |
7.  | 4 | 2 |
8.  | 5 | 1 |
9.  | 6 | 2 |
10. | 7 | 2 |
11. +-----+-----+
```

查找连续出现三次的数字。

```
1.  +-----+
2.  | ConsecutiveNums |
3.  +-----+
4.  | 1                |
5.  +-----+
```

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS LOGS;
4.  CREATE TABLE LOGS ( Id INT, Num INT );
5.  INSERT INTO LOGS ( Id, Num )
6.  VALUES
7.      ( 1, 1 ),
8.      ( 2, 1 ),
9.      ( 3, 1 ),
10.     ( 4, 2 ),
11.     ( 5, 1 ),
12.     ( 6, 2 ),
13.     ( 7, 2 );
```

Solution

```
1.  SELECT
2.      DISTINCT L1.num ConsecutiveNums
3.  FROM
4.      Logs L1,
5.      Logs L2,
6.      Logs L3
7.  WHERE L1.id = L2.id - 1
8.      AND L2.id = L3.id - 1
9.      AND L1.num = L2.num
10.     AND L2.num = L3.num;
```

626. Exchange Seats

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

Description

seat 表存储着座位对应的学生。

```
1.  +-----+-----+
2.  |    id    | student |
3.  +-----+-----+
4.  |     1    | Abbot   |
5.  |     2    | Doris   |
6.  |     3    | Emerson |
7.  |     4    | Green   |
8.  |     5    | Jeames  |
9.  +-----+-----+
```

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。

```
1.  +-----+-----+
2.  |    id    | student |
3.  +-----+-----+
4.  |     1    | Doris   |
5.  |     2    | Abbot   |
6.  |     3    | Green   |
7.  |     4    | Emerson |
8.  |     5    | Jeames  |
9.  +-----+-----+
```

SQL Schema

```
1.  DROP TABLE
2.  IF
3.      EXISTS seat;
4.  CREATE TABLE seat ( id INT, student VARCHAR ( 255 ) );
5.  INSERT INTO seat ( id, student )
```

```
6.  VALUES
7.      ( '1', 'Abbot' ),
8.      ( '2', 'Doris' ),
9.      ( '3', 'Emerson' ),
10.     ( '4', 'Green' ),
11.     ( '5', 'Jeames' );
```

Solution

使用多个 union。

```
1.  SELECT
2.      s1.id - 1 AS id,
3.      s1.student
4.  FROM
5.      seat s1
6.  WHERE
7.      s1.id MOD 2 = 0 UNION
8.  SELECT
9.      s2.id + 1 AS id,
10.     s2.student
11.  FROM
12.     seat s2
13.  WHERE
14.     s2.id MOD 2 = 1
15.     AND s2.id != ( SELECT max( s3.id ) FROM seat s3 ) UNION
16.  SELECT
17.     s4.id AS id,
18.     s4.student
19.  FROM
20.     seat s4
21.  WHERE
22.     s4.id MOD 2 = 1
23.     AND s4.id = ( SELECT max( s5.id ) FROM seat s5 )
24.  ORDER BY
25.     id;
```

