



SQL语言 - SQL题目进阶

1.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
|
+-----+-----+-----+-----+-----+
---+
```

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

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

SQL Schema

```

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' );

```

Solution

```

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

```

2.Swap Salary

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

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' );
```

Solution

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

3.Not Boring Movies

Description

id	movie	description	rating
1	War	great 3D	8.9
2	Science	fiction	8.5
3	irish	boring	6.2
4	Ice song	Fantasy	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。

5	House card	Interesting	9.1
1	War	great 3D	8.9

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 );
```

Solution

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

4.Classes More Than 5 Students

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   |
+-----+
```

SQL Schema

```
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' );
```

Solution

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

5.Duplicate Emails

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 |
+-----+
```

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' );
```

Solution

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

6.Delete Duplicate Emails

Description

邮件地址表：

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

删除重复的邮件地址：

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

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' );
```

Solution

连接:

```
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 );
```

7.Combine Two Tables

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。

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' );
```

Solution

使用左外连接。

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

8. Employees Earning More Than Their Managers

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。

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 );

```

Solution

```

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

```

9. Customers Who Never Order

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       |
+-----+
```

SQL Schema

```
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 );
```

Solution

左外链接

```
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

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

10.Department Highest Salary

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

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' );
```

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;
```

11.Second Highest Salary

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。

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 );
```

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;
```

12.Nth Highest Salary

Description

查找工资第 N 高的员工。

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 );
```

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
```

13.Rank Scores

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

SQL Schema

```

DROP TABLE
IF
    EXISTS Scores;
CREATE TABLE Scores ( Id INT, Score DECIMAL ( 3, 2 ) );
INSERT INTO Scores ( Id, Score )
VALUES
    ( 1, 3.5 ),
    ( 2, 3.65 ),
    ( 3, 4.0 ),
    ( 4, 3.85 ),
    ( 5, 4.0 ),
    ( 6, 3.65 );

```

Solution

```

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

```

14.Consecutive Numbers

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               |
+-----+
```

SQL Schema

```
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 );
```

Solution

```
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;
```

15.Exchange Seats

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

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' );

```

Solution

使用多个 union。

```

SELECT
    s1.id - 1 AS id,
    s1.student
FROM
    seat s1
WHERE
    s1.id MOD 2 = 0 UNION
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
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;
```