175. Combine Two Tables
Easy
2.6K
204
Companies
SQL Schema
Table: Person

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| personId    | int     |
| lastName    | varchar |
| firstName   | varchar |
+-------------+---------+
```

personId is the primary key column for this table.
This table contains information about the ID of some persons and their first and last names.

Table: Address

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| addressId   | int     |
| personId    | int     |
| city        | varchar |
| state       | varchar |
+-------------+---------+
```

addressId is the primary key column for this table.
Each row of this table contains information about the city and state of one person with ID = PersonId.

Write an SQL query to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in any order.

The query result format is in the following example.

Example 1:

Input:
Person table:

```
+----------+----------+-----------+
| personId | lastName | firstName |
+----------+----------+-----------+
| 1        | Wang     | Allen     |
| 2        | Alice    | Bob       |
+----------+----------+-----------+
```
Address table:
```
+-----------+----------+---------------+------------+
| addressId | personId | city          | state      |
+-----------+----------+---------------+------------+
| 1         | 2        | New York City | New York   |
| 2         | 3        | Leetcode      | California |
+-----------+----------+---------------+------------+
```
Output:
```
+-----------+----------+---------------+----------+
| firstName | lastName | city          | state    |
+-----------+----------+---------------+----------+
| Allen     | Wang     | Null          | Null     |
| Bob       | Alice    | New York City | New York |
+-----------+----------+---------------+----------+
```
Explanation:
There is no address in the address table for the personId = 1 so we return null in their city
and state.
addressId = 1 contains information about the address of personId = 2.


```sql
 1•  create database team;
 2•  use team;
 3•  create table person(personid int,firstname varchar(10),lastname varchar(10)) ;
 4•  alter table person add primary key(personid);
 5•  select p.firstName, p.lastName, a.city, a.state from person p right join address a on p.personid=a.personid;
 6•  insert into person values(1,"allen","wang"),(2,"saurav","sharma"),(3,"bob","alice");
 7•⊖ create table address(addressid int primary key, personid int, city varchar(20),state varchar(20),foreign key(personid)
 8   references person(personid));
 9•  select * from address1;
10•  insert into address values(1,2,"New York City", "New York"),(2,3, "Leetcode","California" );
11
12•  select p.firstName, p.lastName, a.city, a.state from person p right join address a on p.personid=a.personid;
```

	1:8

Result Grid | Filter Rows: Search | Export:

firstName	lastName	city	state
saurav	sharma	New York City	New York
bob	alice	Leetcode	California

182. Duplicate Emails
Easy
1.5K
53
Companies
SQL Schema
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.

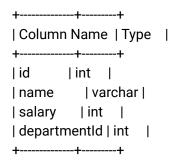Write an SQL query to report all the duplicate emails.

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.

```sql
SELECT a.email
FROM (SELECT email, COUNT(*) as cnt
FROM Person
GROUP BY email) as a
where a.cnt > 1;
```


185. Department Top Three Salaries

Hard
1.4K
197
Companies
SQL Schema
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.

A company's executives are interested in seeing who earns the most money in each of the company's departments. A high earner in a department is an employee who has a salary in the top three unique salaries for that department.

Write an SQL query to find the employees who are high earners 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   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
| 7  | Will  | 70000  | 1            |
+----+-------+--------+--------------+
```

Department table:

```
+----+-------+
| id | name  |
+----+-------+
| 1  | IT    |
| 2  | Sales |
+----+-------+
```

Output:

```
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| IT         | Joe      | 85000  |
| IT         | Randy    | 85000  |
| IT         | Will     | 70000  |
| Sales      | Henry    | 80000  |
| Sales      | Sam      | 60000  |
+------------+----------+--------+
```

Explanation:

In the IT department:

- Max earns the highest unique salary
- Both Randy and Joe earn the second-highest unique salary
- Will earns the third-highest unique salary

In the Sales department:

- Henry earns the highest salary
- Sam earns the second-highest salary
- There is no third-highest salary as there are only two employees

```sql
select e1.Name as 'Employee', e1.Salary
from Employee e1
where 3 >
(
    select count(distinct e2.Salary)
    from Employee e2
    where e2.Salary > e1.Salary
)
;
```