1. Secind HIghest Salary:

```
select coalesce ( (
   select distinct salary
   from Employee
   order by salary Desc
   limit 1 offset 1
), null) as SecondHighestSalary;
```

```
select max(salary) as SecondHighestSalary
from Employee
where salary < (select Max(salary) from Employee);
```

2. Write a solution 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.

```
# Write your MySQL query statement below
SELECT p.firstName, p.lastName, a.city, a.state
from Person p
left join Address a ON p.personId = a.personId;
```

3. Write a solution to find the nth highest distinct salary from the Employee table. If there are less than n distinct salaries, return null.

```
CREATE FUNCTION getNthHighestSalary(N INT) RETURNS INT
BEGIN
  DECLARE offsetVal INT;
  SET offsetVal = N - 1;
  RETURN (
      # Write your MySQL query statement below.
      select distinct salary
      from Employee
      order by salary desc
      limit 1 offset offsetVal
  );
END
```

4. Write a solution to find the rank of the scores. The ranking should be calculated according to the following rules:

The scores should be ranked from the highest to the lowest.

If there is a tie between two scores, both should have the same ranking.

After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

Return the result table ordered by score in descending order.

The result format is in the following example.

```
select score, dense_rank() over (order by score desc) as 'rank'
from scores;
```

5. Find all numbers that appear at least three times consecutively.

Return the result table in any order.

The result format is in the following example.

```
select distinct num as ConsecutiveNums
from (
   select num,
      Lead(num, 1) over (order by id) as next1,
      Lead(num, 2) over (order by id) as next2
   from logs
) as temp
where num = next1 and num = next2;
```

6. Write a solution to find the employees who earn more than their managers.

Return the result table in any order.

The result format is in the following example.

Input:

Employee table:

```
+----+-------+--------+-----------+
| id | name  | salary | managerId |
+----+-------+--------+-----------+
```

| 1 | Joe | 70000 | 3 |

| 2 | Henry | 80000 | 4 |

| 3 | Sam | 60000 | Null |

| 4 | Max | 90000 | Null |

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

Output:

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

| Employee |

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

| Joe |

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

```
select e1.name as 'Employee'
from Employee e1
Join Employee e2 on e1.managerId = e2.id
where e1.salary > e2.salary;
```

7. Find the duplicate rows

```
SELECT p.*
FROM Person p
JOIN (
  SELECT name, email
  FROM Person
  GROUP BY name, email
```

```
  HAVING COUNT(*) > 1
) dup
ON p.name = dup.name AND p.email = dup.email;
```

8. Write a solution to find all customers who never order anything.

Return the result table in any order.

The result format is in the following example

Input:

Customers table:

```
+----+-------+
| id | name  |
+----+-------+
| 1  | Joe   |
| 2  | Henry |
| 3  | Sam   |
| 4  | Max   |
+----+-------+
```

Orders table:

```
+----+------------+
| id | customerId |
+----+------------+
| 1  | 3          |
| 2  | 1          |
+----+------------+
```

```
select name as Customers
from Customers c
where c.id not in (
    select o.customerId
    from Orders o
)
```

9.