Documentation: Finding the Second Highest Distinct Salary in an Employee Table

Problem Statement

The task is to write an SQL query that retrieves the second highest distinct salary from the Employee table. If there is no second highest salary, the query should return NULL.

Table Schema

- **Table Name:** Employee
- **Columns:**
  - ***id (int):*** The unique identifier for each employee. This is the primary key.
  - ***salary (int):*** The salary of the employee.

Each row in the Employee table represents an individual employee and their corresponding salary.

Requirements

1. Primary Objective:

- To find the second highest distinct salary from the Employee table.
- The query should return a single column result set with the alias SecondHighestSalary.

2. Handling Edge Cases:

- If there is only one unique salary in the table, the query should return NULL as there isn't a second highest distinct salary.
- If the Employee table is empty, the result should also be NULL.

2. **Output Format:**
   - ***The result should be a single column table:***

SecondHighestSalary
second_highest

Replace [second_highest] with the actual second highest salary or NULL if it does not exist.

Approach to Solution

1. **Understanding Distinct Salaries:**
   - The term "distinct salary" refers to the unique salary values present in the Employee table. We must ensure that the query takes into account only distinct salaries to accurately determine the second highest value.

2. **Ordering and Limiting Results:**
   - To find the second highest salary, it is crucial to first sort the distinct salary values in descending order (from highest to lowest).
   - Once the salaries are sorted, the second highest value can be obtained by selecting the salary positioned second in this sorted list.

3. **Utilizing SQL Functions and Keywords:**

- ***DISTINCT:*** This keyword is used to ensure that only unique salary values are considered.
- ***ORDER BY:*** This clause is used to sort the salary values in descending order.
- ***LIMIT and OFFSET:*** *These clauses help in selecting the desired position in the sorted list:*
  - LIMIT 1 specifies that only one result should be returned.
  - OFFSET 1 skips the first row (the highest salary), thereby fetching the second row (the second highest salary).

4. **Null Handling:**
   - In cases where there is no second highest distinct salary, the SQL query naturally returns NULL. This occurs because the query is designed to return a specific position in the result set, and if that position does not exist, SQL defaults to NULL.

Example 1:

Input:

id	salary
1	100
2	200
3	300

Output:

SecondHighestSalary
200

Explanation: The distinct salaries in descending order are 300, 200, and 100. The second highest salary is 200.

Example 2:

Input:

id	Salary
1	100

Output:

SecondHighestSalary
null

Explanation: There is only one distinct salary (100). Since there is no second highest distinct salary, the output is NULL.

Example 3:

Input:

(Empty Table)

Output:

SecondHighestSalary
null

Explanation: The Employee table is empty, so there are no salary values to evaluate. Thus, the result is NULL.

Conclusion

- The SQL query is designed to efficiently and accurately retrieve the second highest distinct salary from the Employee table while accounting for edge cases such as an empty table or the absence of a second distinct salary. By leveraging SQL functions and clauses such as DISTINCT, ORDER BY, LIMIT, and OFFSET, the solution is both robust and straightforward.