Database and SQL Answers

1. Queries number 1, 2 and 3 are correct. Here’s why:

* Query 1: SELECT DISTINCT(salary) FROM emp ORDER BY salary DESC LIMIT 1 OFFSET 1;
  + This selects the unique salaries (removes duplicates), then orders them in descending order, then gets the salary after the first and ideally, this will be the next highest salary.
* Query 2: SELECT MAX(salary) FROM emp WHERE salary < (SELECT MAX(salary) FROM emp);
  + This query selects the next max salary which is the next highest salary.
* Query 3: SELECT salary FROM (SELECT DISTINCT salary FROM emp ORDER BY salary DESC LIMIT 2) AS emp ORDER BY salary LIMIT 1;
  + This query first selects the first two highest, unique salaries, then orders the two salaries in ascending order and selects the first salary, and that is the second highest salary.
* Query 4: SELECT DISTINCT salary FROM (SELECT salary FROM emp ORDER BY salary DESC LIMIT 2) AS emp ORDER BY salary LIMIT 1;
  + The subquery here does not remove the duplicated salaries by using the “DISTINCT” keyword, so the first two highest salaries may not be the same, therefore, the selected lower salary will not be the second highest salary.
* Query 5: SELECT salary FROM emp ORDER BY salary DESC OFFSET 1 LIMIT 1;
  + This query does not remove duplicated salaries; therefore, the selected salary may not be the second highest.

1. SELECT g.yr, c.country

FROM games g

JOIN city c ON g.city = c.name

ORDER BY g.yr;

1. LEFT JOIN Case:

This will include all the rows from the left table and the matched rows from the right table. If there is no match, the result will be a NULL from the right table.

In this case, “games” is the left table and “city” is the right table.

Using this query for an example:

SELECT g.yr, g.city, c.country

FROM games g

LEFT JOIN city c ON g.city = c.name;

The result will be:

|  |  |  |
| --- | --- | --- |
| **yr** | **city** | **country** |
| 1896 | Athens | Greece |
| 1948 | London | UK |
| 2004 | Athens | Greece |
| 2008 | Beijing | China |
| 2012 | London | UK |

RIGHT JOIN Case:

This will include all the rows from the right table and the matched rows from the left table. If there is no match, the result will be a NULL from the left table.

Still assuming “games” is the left table and “city” is the right table.

Using this query for an example:

SELECT g.yr, g.city, c.country

FROM games g

RIGHT JOIN city c ON g.city = c.name;

The result will be:

|  |  |  |
| --- | --- | --- |
| **yr** | **city** | **country** |
| 1896 | Athens | Greece |
| 1948 | London | UK |
| 2004 | Athens | Greece |
| 2008 | Beijing | China |
| 2012 | London | UK |
| NULL | Sydney | Australia |

1. SELECT userId, AVG(duration) AS AverageDuration

FROM sessions

GROUP BY userId

HAVING COUNT(id) > 1;