1. Union과 union all 차이

* UNION은 UNION DISTINCT와 동일한 작업을 하기 때문에 중복되는 레코드를 제거했음을 알 수 있다.
* 하지만, UNION ALL의 경우에는 별도의 중복 제거 과정을 거치지 않고 그냥 결과를 내려준다.

1. 조인 종류

INNER JOIN은 키 값이 있는 테이블의 컬럼 값을 비교 후 조건에 맞는 값을 가져오는 것이다. 간단히 말하면 서로 연관된 내용만 검색하는 조인 방법이다.

CROSS JOIN은 Cartesian Product(카디션 곱)이라고도 하며 조인되는 두 테이블에서 곱집합을 반환한다. 즉, 두 번째 테이블로부터 각 열과 첫 번째 테이블에서 각 열이 한번씩 결합된 열을 만들 것이다. 예를 들어 m열을 가진 테이블과 n열을 가진 테이블이 교차 조인되면 m\*n 개의 열을 생성한다. 그래서 테이블의 각 값을 연결하여 테이블 행의 수를 모두 곱한 값만큼 만들어진다.

LEFT OUTER JOIN은 조인문의 왼쪽에 있는 테이블의 모든 결과를 가져온 후 오른쪽 테이블의 데이터를 매칭하고, 매칭되는 데이터가 없는 경우 NULL을 표시한다.

RIGHT OUTER JOIN은 조인문의 오른쪽에 있는 테이블의 모든 결과를 가져온 후 왼쪽의 테이블의 데이터를 매칭하고, 매칭되는 데이터가 없는 경우 NULL을 표시한다.

Full Outer Join은 LEFT OUTER JOIN과 RIGHT OUTER JOIN을 합친 것이다.

양쪽 모두 조건이 일치하지 않는 것들까지 모두 결합하여 출력한다.

1. select case when null = null then 'Yup' else 'Nope' end as Result; 결과

This query will actually yield “Nope”, seeming to imply that null is not equal to itself! The reason for this is that the proper way to compare a value to null in SQL is with the is operator, not with =.

Accordingly, the correct version of the above query that yields the expected result (i.e., “Yup”) would be as follows:

select case when null is null then 'Yup' else 'Nope' end as Result;

This is because null represents an unknown value. If you don’t know the value, you can’t know whether it equals another value, so = null is always assumed to be false.

1. What is wrong with this SQL query? Correct it so it executes properly.

SELECT Id, YEAR(BillingDate) AS BillingYear

FROM Invoices

WHERE BillingYear >= 2010;

The correct query should be:

SELECT Id, YEAR(BillingDate) AS BillingYear

FROM Invoices

WHERE YEAR(BillingDate) >= 2010;

Write a SQL query to find the 10th highest employee salary from an Employee table. Explain your answer.

Given the following tables:

sql> SELECT \* FROM runners;

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

| id | name |

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

| 1 | John Doe |

| 2 | Jane Doe |

| 3 | Alice Jones |

| 4 | Bobby Louis |

| 5 | Lisa Romero |

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

sql> SELECT \* FROM races;

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

| id | event | winner\_id |

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

| 1 | 100 meter dash | 2 |

| 2 | 500 meter dash | 3 |

| 3 | cross-country | 2 |

| 4 | triathalon | NULL |

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

What will be the result of the query below?

SELECT \* FROM runners WHERE id NOT IN (SELECT winner\_id FROM races)

Explain your answer and also provide an alternative version of this query that will avoid the issue that it exposes.

Given these contents of the Customers table:

Id Name ReferredBy

1 John Doe NULL

2 Jane Smith NULL

3 Anne Jenkins 2

4 Eric Branford NULL

5 Pat Richards 1

6 Alice Barnes 2

Here is a query written to return the list of customers not referred by Jane Smith:

SELECT Name FROM Customers WHERE ReferredBy <> 2;

What will be the result of the query? Why? What would be a better way to write it?

Table is as follows:

| **ID** | **C1** | **C2** | **C3** |
| --- | --- | --- | --- |
| 1 | Red | Yellow | Blue |
| 2 | NULL | Red | Green |
| 3 | Yellow | NULL | Violet |

Print the rows which have ‘Yellow’ in one of the columns C1, C2, or C3, but without using OR.