<데이터베이스 실습 복습 정리>

160825~160907(SELECT문~문자열함수)
(문장은 간략하게 작성하는 목적상 소문자로 작성하겠습니다. 구분이 필요할 때만 대문자를 작성합니다.)

1. Select 문 제대로 써먹기 at 160831

1) 테이블의 구조 파악하는 문장
desc[ribe] employees;
// 이처럼 작성하게 된다면 나오는 정보는 컬럼 이름, 널 여부, 데이터 타입 이렇게 나오게 된다.

2) 컬럼들을 조회하는 문장
select *
from employees;
// 이처럼 작성하게 된다면 employees에 있는 모든 컬럼들에 관련되어서 튜플들이 출력된다.

select first_name, salary, phone_number
from employees;
// 이처럼 작성하게 된다면 사용자가 원하는 컬럼들만 관련되어서 튜플들이 출력된다.

3) 컬럼 이름 대신 임의 이름으로 출력
select first_name "Name", phone_number AS "Tel", salary Salary
from employees;
// 이처럼 작성하게 된다면 first_name 컬럼은 Name으로 설정이 되고, phone_number는 Tel로 지정이 된다. 물론 큰 따옴표로 굳이 안해도 상관없다. 허나 Salary는 SALARY로 임의 이름이 바뀌게 된다. 그래서 대/소문자를 구분하는 역할을 하기 위해서는 큰 따옴표로 감추도록 하자.(물론 그냥 컬럼 이름도 이와 같은 원리이다.)

4) 컬럼이 아닌 리터럴 문자를 출력하기
select employee_id, 'Nicer~'
from employees;
// 이처럼 작성하게 되면 employee_id의 개수만큼 Nicer~가 여러 개 출력하게 된다. 허나 주의할 점은 작은 따옴표로 가려야 쓸 수 있다는 점을 명심하고, 컬럼 이름은 'NICER~'로 지정이 된다는 점을 명심하자.
참고로 SQL 본래 문자는 대/소문자까진 구분 안하지만, 작은 따옴표 내부에 있는 문자열들은 구분하니깐 주의하길 바란다.(문자열 Dok2는 문자열 dok2과 엄연히 다르다.)

5) 연결 고리 연산자 ||
select last_name||'의 월급은'||salary||'달러~'
from employees;
|| 는 데이터와 문자열을 연결하는 연산자이다. 지난번 퀴즈에서 '달러~' 뒤에 ||를 작성하는 실수를 했지만 중간고사에서 이런 실수는 절대 하지 않기를 바란다.

6) 중복된 값 제거하는 distinct
select **distinct** job_id
from employees;
distinct 키워드는 이 문장에서 직원들이 속한 job_id을 중복이 없게끔 출력하는 문장이다. 그렇지만 우리가 배우는 버전은 정렬을 알아서 써야 되고, 무조건 컬럼 시작할 때 밑줄친 부분에다만 쓸 수 있으니 주의하자.

7) Where 연산자 이용하기

7-1) 부등호 활용하기
select first_name, salary
from employees
where salary>=4000;
이처럼 수치에서는 부등호를 이용해서 조건을 형성할 수 있다. 이 문장은 쉽게 이야기해서 봉급이 4000 이상인 사람들을 출력하는 문장이다. 부등호는 다음과 같이 쓰면 된다.
컬럼(> : 초과, >= : 이상, < : 미만, <= : 이하, = 같다, <> != : 다르다)

7-2) 산술 연산자 활용
select first_name, salary*3 "3Month_Salary"
from employees
where department_id=60;
이 문장에서 쓰인 연산자 중 *이 보일 것이다. 이는 쉽게 말해 곱셈이다. +, -, *, / 그대로 적용이 되니 참고하자.(허나 나머지 출력인 %는 안 된다는 점은 알고 있자.)

7-3) Between A and B 연산자
select first_name, salary
from employees
where employee_id Between 120 and 145;
Between A and B 연산자는 직원 번호가 120번과 145번에 속하는 모든 직원들의 이름과 봉급을 출력하게 된다. 햇깔리면 안 되는 점은 120번~145번 사이가 아니라 120번도 출력되고 145번도 출력이 된다. 쉽게 이야기해서 다음 문장과 같은 의미로 해석하기 바란다. 하지만 아래 문장보다는 처리 속도가 의외로 느린 키워드가 Between이라는 건 숙지하고 있자.
select first_name, salary
from employees
where employee_id>=120 AND employee_id<=145;

7-4) IN 연산자
select first_name, salary
from employees
where job_id IN('ST_CLERK', 'ST_MAN');
문자열을 이용해서 출력하게끔 하려면 작은 따옴표는 무조건 달아줘야지 문자열로 해석하게 된다.
select first_name, salary
from employees
where department_id IN(60, 70, 80);
물론 숫자는 그냥 입력하면 알아서 출력하게 된다. 또한 IN 연산자는 처리 속도도 매우 양호해서 많이 쓰인다는 점을 알아두자.

7-5) LIKE 연산자
여기서 쓰는 문장인 % 연산자는 글자 수는 제한이 없고 글자 종류가 뭐가 됐든 상관이 없다. 가령 Like '강%'을 입력하면 결과 값은 강, 강남역, 강변, 강서구... 이런 식으로 나오게 된다.
반면 _연산자는 글자 수는 한 글자만(Must 있어야 한다.) 들어 올 수 있고, 글자 종류는 어떻든 상관이 없다. 만일 Like '구_역'을 입력하게 된다면 결과 값은 구성역, 구의역, 구로역, 구산역... 이런 식으로 출력이 된다. 조금만 응용해서 Like '_원%'을 입력하게 되면 결과 값은 수원, 일원역, 노원구... 이런 식으로 나오게 된다.
다음 문장을 살펴보도록 하자.

Select first_name, salary
from employees
where job_id Like 'SA%'
이처럼 입력하게 된다면 job_id가 SA_MAN, SAC_Woman 등등 출력이 된다. 허나 가령 예를 들어 SA_가 포함되어 출력하는 경우에는 어떻게 해야 하는가?? 애석하게도 SA_%라고 입력하면 SAC_... 이런 것들도 출력이 되니 다음과 같이 바꿔서 작성해 보도록 하자.
Select first_name, salary
from employees
where job_id Like 'SA_%' escape '\';
이처럼 작성하게 된다면 SA_만 포함하게끔 되어 결국 원하는 값을 출력하는 해피엔딩으로 종결된다. 이 문장에서 \가 포함이 되어 있다는 걸 깨닫지만, escape를 이용해서 지워주고 SA_를 포함하는 문장에 대해 출력하게끔 해준다. 시험에 나올 듯 하니 알아둡시다.

7-6) Is NULL/Is not NULL 연산자
select first_name, salary*commission_pct "bonus"
from employees
where commission_pct is Not Null;
이처럼 문장을 작성하게 된다면 보너스 비율이 존재하는 값들의 튜플들만 출력하게 해준다. 그래서 is not Null 연산자는 값이 존재하면 모든 튜플이 출력하게끔 해주는 연산자라는 점을 알아두자.

select first_name, salary
from employees
where commission_pct is Null;
허나 보너스 비율이 없는 직원들이 있을 것이다. 재미있는 점은 C언어에서 null를 0으로 대입해서 써먹었지만 데이터베이스는 null이 0이나 공백이랑 엄연히 다른 점을 깨달아두자. 그래서 값이 아예 없는 튜플들만 출력해주는 역할을 하는 연산자는 is Null이라는 점을 알아두자.

7-7) AND, OR, NOT 연산자
select employee_id, first_name
from employees
where salary>=10000 And department_id=50;
이 문장은 쉽게 이야기해서 봉급이 만 달러 이상이고, 부서 번호가 50인 튜플에 대해 출력하는 것이다. 두 조건을 모두 만족하는 튜플을 출력하니깐 어느 하나라도 거짓이면 출력을 안 한다는 점을 알아두자.

select employee_id, first_name
from employees
where salary>=10000 Or department_id=50;
이 문장은 봉급이 만 달러 이상이거나 부서 번호가 50인 튜플에 대해 출력하는 것이다. 두 조건 중 아무거나 속해도 출력하게끔 해준다.

select employee_id, first_name
from employees
where Not department_id=50;
이 문장은 쉽게 이야기해서 부서 번호가 50번이 아닌 직원들의 튜플이 출력되는 이야기이다. Not 연산자도 의외로 많이 쓰이니 알아두도록 하자.

8) Order by 문장
Select first_name, salary, department_id
from employees
where salary>=5000
order by first_name ASC;
이처럼 문장을 작성하면 first_name에서 알파벳이 커지는 순서(A~z순서대로. 대문자우선 // 숫자는 0부터 커지는 순서 // 날짜는 옛날부터 지금으로)대로 출력이 된다. 물론 오름차순이라면 ASC를 굳이 쓸 필요까지는 없다. 허나 내림차순은 DESC를 뒤에 써줘야 한다.
Select first_name, salary, department_id
from employees
where salary>=5000
order by first_name DESC;
이처럼 작성하게 되면 내림차순으로(z~A순서대로. 소문자 우선 // 숫자는 큰 값부터 작아지는 순서 // 날짜는 최근에서 옛날로) 출력이 된다. 만일 오름차순과 내림차순을 같이 쓰게 되면 다음 문장을 살펴보면 된다.
// 1
Select first_name, department_id, salary "봉급"
from employees
where salary>=5000
order by department_id, 봉급 DESC;
다음과 같이 작성을 하게 되면 department_id는 커지는 순서대로 출력하되, department_id의 값이 같다면 봉급은 작아지는 순서대로 출력하게끔 해준다. 더욱 재미 있는 건 AS를 이용해 칼럼 이름을 대신한 봉급을 그대로 써도 작동하는데 문제는 없다.(Order By만 된다.)
// 2
Select first_name, department_id, salary "봉급"
from employees
where salary>=5000
order by 2, 3 DESC;
물론 더욱 재미있는 건 사용자가 원하는 칼럼에서 칼럼 번호는 first_name부터 1번, department_id는 2번 이런 식으로 칼럼 번호가 지정이 된다. 그래서 order by에서는 칼럼 값을 이용해서 정렬하는 방법이 있다. 문제는 1번과 같으니 참조하자.

---

2. 집합 연산자 at 160905

집합 연산자는 방금 And, Or, Not 연산자를 다뤘을 것이다. 허나 한 문장 내에서 끝나지 않고 여러 문장들에 관련된 집합 연산을 해서 출력하는 건 생각도 못해봤을 것이다. 그래서 여러 문장을 다룰 때 쓰는 집합 연산자를 알아둘 필요가 있다는 점을 밝힌다.

1) Union, Union All 연산자(합집합 개념)
우리가 책에 나온 조건을 이용해서 쉽게 알아보도록 하자. 책 45페이지에서 참고하면 (조건1)은 봉급이 10000달러를 넘고, 부서ID가 80번인 직원의 부서ID와 봉급을 뽑아내는 것이고, (조건2)는 봉급이 10000달러를 넘고, 부서ID가 90번인 직원의 부서ID와 봉급을 뽑아내는 것이다. 조건 1의 결과는 3개이고, 조건 2의 결과는 8개가 나오는 것이다. 그렇지만 Union과 Union All의 차이에 대해 숙지해둘 필요가 있다. 구분하기 쉽게 설명하면 Union은 그냥 중복 그런 거 없이 나오는 것이고, Union All은 중복되든 나발이든 모두 나오는 개념이다. 그래서 Union 문장과 Union All 문장을 통해서 각 개념에 대해 살펴보도록 하자.

1-1) Union 연산자
```
select department_id, salary
from employees
where salary>10000 and department_id=90
Union
select department_id, salary
from employees
where salary>10000 and department_id=80;
```
이처럼 작성을 하게 된다면 11개의 튜플이 나올거라 생각하는 사람들이 많지만, 애석하게도 8개만 나오게 된다. 그래서 이 문장은 쉽게 이야기해서 다음 문장과 같다고 생각하면 되겠다.
```
select distinct department_id, salary
from employees
// 1 where salary>10000 and (department_id=80 or department_id=90);
// 2 where salary>10000 and department_id IN(80, 90);
```

1-2) Union All 연산자
```
select department_id, salary
from employees
where salary>10000 and department_id=90
Union All
select department_id, salary
from employees
where salary>10000 and department_id=80;
```
이처럼 작성을 하게 된다면 11개의 튜플이 최종적으로 출력하게 된다. All은 쉽게 이야기해서 중복되어도 출력하게끔 해주는 개념으로 생각하면 된다. 그래서 이는 다음 문장과 같다고 생각하면 된다.
```
select department_id, salary
from employees
// 1 where salary>10000 and (department_id=80 or department_id=90);
// 2 where salary>10000 and department_id IN(80, 90);
```

2) Intersect 연산자(교집합 개념)
Intersect 연산자는 And 연산자와 같은 원리로서 쓰는 함수이다. 반대로 (조건1)은 부서ID가 50번이고, (조건2)는 봉급이 3000달러 이하인 조건이다. 그래서 다음과 같이 Intersect 문장을 이용해서 출력해보도록 하자.
```
SELECT employee_ID
from employees
where salary<=3000
INTERSECT
SELECT employee_ID
from employees
where department_ID=50;
```
이처럼 문장을 출력을 하게 된다면 22개가 출력하게 된다. Intersect는 중복된 값을 출력하는 개념이 절대 아니란 점을 명시하자. 물론 이 문장은 다음과 같이 대체할 수 있다는 점도 알아두면 좋다.
```
Select employee_id
from employees
where (salary<=3000 and department_id=50);
```

3) Minus 연산자(차집합 개념)
Minus 연산자는 큰 집합 개념에서 작은 집합 개념을 빼내는 연산자라고 생각하면 된다. 다음 문장을 살펴보자.
```
SELECT employee_ID, last_name
from employees
where salary<=3000
MINUS
SELECT employee_ID, last_name
from employees
where department_ID=50;
```
다음과 같이 작성을 하게 된다면 봉급이 3000이하인 사람들 중에서 부서ID가 50번인 사람을 제외시킨다. 다음과 같은 문장은 And와 not로서 대체가 가능하다고 생각하면 된다. ($A-B=A\cap B^c$ 인 개념을 이용하면 된다.)
```
SELECT employee_ID, last_name
from employees
where salary<=3000 and not department_ID=50;
```

(참고) 허나 집합 연산자를 쓰게 된다면 모든지 정렬이 되어 나오기 때문에 처리 속도가 그닥 빠른 편은 아니라는 점은 숙지하길 바란다. 또한 Union, Minus, Intersect 연산자는 컬럼이 2개 이상인 경우에는 중복을 없애는 역할을 한다는 점도 숙지해두길 바란다.
또한 집합 연산자를 이용하는 경우에는 컬럼의 수를 맞춰서 작성해야 한다. 안 그러면 출력하는 것에 대하여 오류가 걸리게 되니 주의하도록 하자.

---

3. 문자열 함수 at 160907

1) 함수의 종류
함수는 단일 행 함수/복수 행 함수 2가지로 나뉘게 된다. 단일 행 함수는 한 튜플에 있는 데이터를 다루는 함수이다. 허나 복수 행 함수는 조건에 해당되는 튜플들에 관련해서 그룹으로 묶어서 쓰는 함수라고 생각을 하면 된다. 단일 행 함수의 종류에는 문자 함수, 숫자 함수, 날짜 함수, 변환 함수(묵시적 데이터 변환/명시적 데이터 변환), 일반 함수 5가지로 쉽게 나뉜다. 우선 9월 7일에 배운 문자 함수를 기준으로 해서 자세히 숙지해보자.

2) Initcap() 함수
Initcap는 Initial Capital(앞 글자는 대문자로)의 줄임말이라고 숙지하면 쉽게 외워진다. 이는 문자열이나 칼럼에서 첫 번째 글자가 대문자로 쓰여지게끔 참조하는 함수이다.(national을 예로 들면 National로 참조하게 해준다.)Initcap함수는 다음과 같이 쓴다.
INITCAP(문자열 or 칼럼이름)
이는 우리 말에서는 쓰이는 일이 없겠다만 영어권 국가에서 주로 쓰인다는 점을 숙지해두자.

3) Upper(), Lower() 함수
Upper() 함수는 말 그대로 문자열이나 함수를 대문자로, Lower() 함수는 문자열이나 컬럼을 소문자로 보이게끔 참조하는 함수이다. 이를 이용하는 방법은 Initcap 함수처럼 쓰면 된다.

4) length(), lengthb() 함수
우리가 가령 예를 들어 '도끼더콰이엇'이란 단어가 있다고 하자. 이를 length() 함수와 lengthb() 함수에 적용시켜 보겠다.
length('도끼더콰이엇') -> 6
length는 빈 칸을 포함해서 문자열에 들어간 알파벳/한글 1자/숫자 개수라고 생각하면 된다.
lengthb('도끼더콰이엇') -> 18
lengthb는 lengthByte의 개념으로 생각하면 된다. 그러나 sql에서는 한글 한 글자를 3바이트로 인식을 한

다는 점을 고려해야 한다. 물론 영어는 자기가 만들었으니 1바이트로 인식을 할 것이다(NLS_Charset가 AL32UTF8, UTF8로 저장되었다면 한글 한 글자가 3바이트인 점으로서, 이의 설정에 따라서 2바이트도 될 수 있다.)

5) SUBSTR() 함수, SUBSTRB() 함수
Substr는 substring의 줄임말로서 문자열에서 몇 개를 뽑아가겠다는 의미로 생각하면 된다.
Substr(문자열/컬럼, 시작위치, 골라낼 글자수)
Substr('808베이스소리', 4, 3) // 1
Substr('808베이스소리', -3, 3) // 2
1 : 시작 위치가 보통 C언어나 자바에서 0번째부터로 인식하는 사람들이 많은데 SQL에서는 1번째부터가 바로 시작이다.(즉 위 문장에서 8이 첫 번째로 의식해야 한다.) 그래서 왼쪽에서 4번째 문자인 '베'에서부터 3글자를 뽑아내니깐 결국 출력되는 것은 베이스가 된다.
2 : 물론 시작위치에 음수가 적힐 수가 있다. 이는 뒤에서 3번째 문자부터 시작해서 그 문자부터 시작하여 3글자까지 출력하는 개념으로 숙지해두면 좋다. 그래서 결국 출력되는 문자는 스소리가 된다.

물론 SUBSTRB() 함수도 있지만 이는 한글인 경우엔 한 글자의 바이트를 고려해서 써야 하기 때문에 다음과 같이 쓰면 큰 도움이 된다.
Substrb('808베이스소리', 4, Lengthb('나비야'))
그러면 lengthb에서 한글 바이트 수를 측정하고 그만큼 출력해서 결국에 방금 위처럼 베이스가 나오게 된다. 또한 참고로 substr는 범위가 벗어나도 값은 그 문자열 끝까지 나오고 별 다른 문제가 없다는 것도 숙지해두자.

6) INSTR() 함수
이 함수는 In String의 줄인 말로 생각하면 쉽다. 그래서 이 문자열 안에 특정 글자가 몇 번째에 있는지에 대해 출력하는 것이다.
Instr(문자열/컬럼, 찾는 글자, 시작 위치, 몇 번째(기본 값은 1))
이게 뭔 개소리인지 모를 것이다. 쉽게 이야기해서 '그대 기억이 지난 사랑이' 문장에서 이가 2개가 들어가 있을 것이다. 그러나 이가 몇 번째에 있는지 출력하기 애매해질 때 첫 번째 이는 몇 번째에 있는지와 두 번째 이는 몇 번째에 있는지에 대해 출력을 하는 것이다. 또한 시작위치는 양수와 음수로서 나뉠 수가 있는데 음수이면 뒤에서부터 처음인지 두 번째인지에 대해 참고하니깐 햇깔리는 일이 없도록 하자.

Instr('그대 기억이 지난 사랑이', '이', 1, 1) -> 6
이처럼 작성하게 되면 첫 번째 글자에서 시작하여 첫 번째 이는 어디에 있는지에 대하여 출력을 하는 것이다. 그래서 좌측부터 순차적으로 탐색을 하면 6번째에 이가 있어서 6이 나오게 된다.

Instr('그대 기억이 지난 사랑이', '이', 1, 2) -> 13
첫 번째 글자에서 시작하여 두 번째 이는 어디에 있는지에 대해 출력을 하라는 뜻이다. 그래서 순차 탐색을 지나면 13이 출력된다.

Instr('그대 기억이 지난 사랑이', '이', -1, 1) -> 13
이는 맨 뒤의 글자인 이부터 시작해서 뒤에서 첫 번째 이를 찾는 문장이다. 그래서 출력되는 값은 뒤에서 맨 처음에 발견된 13이다.

Instr('그대 기억이 지난 사랑이', '이', -1, 2) -> 6
또한 이 문자는 맨 뒤의 글자인 이부터 시작해서 뒤에서 두 번째 이를 찾는 문장이다. 그래서 출력되는 값은 6번째에 존재하게 되어 6이 출력하게 된다.

7) LPad(), RPad() 함수
키패드라는 말이 버튼(혹은 키)들이 가득 채워있어서 키들의 모임이라는 뜻이 있듯이 Pad는 채운다는 뜻이 있다. 그래서 LPad() 함수와 RPad() 함수는 다음과 같이 쓰면 된다.
LPad(칼럼/문자열, 채울 문자 수, 채울 문자)
RPad(칼럼/문자열, 채울 문자 수, 채울 문자)
가령 여러분들이 'Dok2'라는 4글자 문자를 썼다고 하자. 그래서 8글자로 완성을 하되 남는 공간을 '*'로서 채우고 싶을 때 이 문자를 쓰면 된다.(참고로 문자 수는 영어로 기준을 한다.)
RPad('Dok2', 8, '*') 이처럼 작성하면
Dok2**** 가 출력하게 된다.
LPad('Dok2', 8, '*') 이처럼 작성하면
****Dok2 가 출력하게 된다.
물론 조금만 두 함수를 잘 활용하면 이처럼 쓸 수 있다.
LPad(RPad('Dok2', 8, '*'), 12, '*')
그러면 출력값은 ****Dok2****가 나오게 된다.

8) LTrim(), RTrim() 함수
Trim은 무언가를 없애는 뜻이다. Trim 함수의 구성은 다음과 같이 되어 있다.
LTrim(문자열/칼럼, '제거할 문자')
RTrim(문자열/칼럼, '제거할 문자')
이는 단어들보다는 글자(알파벳) 같은 데에 쓰는 것을 중점으로 하는 것이 도움이 된다.(물론 단어도 되긴 하는데 차라리 substr를 이용하는 걸 추천한다.) 가령 예를 들어 이름이 Book, boy, Puppy, Lady 단어 내용이 담긴 컬럼을 word라고 하면 뒤에 y를 없애는 실험을 해보겠다.
RTrim(word, 'y')
그러면 뒤에 y가 포함된 모든 단어는 Book, bo, Lad, Pupp로 바뀌게 된다.
허나 반대로 LTrim 함수를 이용해서 앞에 B를 없애는 문장을 써보겠다.(RTrim을 실행 안했다고 가정)
LTrim(word, 'B')
그러면 ook, boy, Puppy, Lady 이런 식으로 저장이 된다. 여기서 대문자 B만 없애라는 거지 소문자 b를 없애라는 거로 이해했다면 문자열에선 대/소문자를 구분한다고 생각을 해야 한다.

9) Replace() 함수
이는 어디가든 많이 쓰는 함수이다. 다른 건 몰라도 이건 꼭 숙지하고 넘어가길 바란다.
함수의 구성은 다음과 같다.
Replace(문자열/컬럼, '문자1', '문자2')
Replace() 함수는 혼자 쓰면 뭔가 재미없는 함수이다. 그래서 방금 배운 substr() 함수까지 병용해서 써보도록 하겠다. 우선 일반적인 이용방법이다.

9-1) 그냥 이용하기
voac, acmping, acmera (word 칼럼에 이것이 저장되어 있다고 가정하자.)
word 컬럼에 ac가 포함되었다면 ca로 바꿔주도록 하는 문장을 작성하겠다. 이건 그냥 평범한 방법이다.

Replace(word, ‘ac’, ‘ca’);
이처럼 작성하게 되시면, 결국 voca, camping, camera 이런 식으로 정상적으로 출력이 된다. 허나 이렇게 쓴다면 뭔가 부족하다는 느낌이 들 것이다. 그래서 방금 배운 substr 함수를 적절히 이용해보자.

9-2) substr 함수를 병용해서 사용자 이름 가리기
Replace(last_name, substr(last_name, 2, 3), ‘***’)
이렇게 하면 가령 이름이 Robot라면 R***t로 정상적으로 출력이 될 것이다. 이는 어느 이벤트에서 당첨되었다면 가운데 이름을 문자로 채우는 형식으로(강*성 이런 식으로) 주로 쓰이기 때문에 이렇게 응용하는 방법은 꼭 알고 넘어가면 좋겠다.

(참고) 지난번에 설명한 Concat라는 함수가 있는데 이는 2개의 매개변수를 통해서 문자열을 합쳐주는 역할을 한다. 본래 이는 MySQL에서 쓰이는 함수인데 오라클 SQL에서도 문제없이 작동되니 알아두는 것도 나쁘지 않다.
Select Concat(‘내가 ’, ‘망할 거 같애?’) “DOK2-내가”
from Dual;