



SQL, PL/SQL(ORACLE)

2. 내부함수 그룹화 그룹조건

< 함수(Function) >

(1) 함수(function) 란? (처리/반환)

어떠한 일을 수행하는 기능으로써 주어진 인수(argument)를 재료로 처리를 하여 그 결과를 반환하는 일을 수행

(2) 함수 기능의 구체적 표현

- 1) Data에 대한 계산
- 2) Data를 다른 형태로 변환
- 3) Data의 결과를 출력

(3) 함수의 종류

- 1) 단일행 함수
-> 하나의 행(row)당, 하나의 결과값을 반환하는 함수
- 2) 복수행 함수
-> 여러개의 행당, 하나의 결과값을 반환하는 함수



(4) 단일행 함수

1) 문자 함수

<1> CHR(아스키코드)

SQL> select chr(65) from dual;

SQL> select 25*25 from dual;

// dual TABLE은 실제로는 존재하지 않는 DUMMY 테이블로

// 간단히 출력해보고자 할때 사용한다.

<2> CONCAT(컬럼명, '붙일문자열')

SQL> select first_name, job_id from employees;

SQL> select first_name || ' is a ' || job_id from employees;

SQL> select CONCAT(first_name, ' is a '), job_id from employees;

<3> INITCAP('문자열') // 시작문자를 대문자로

SQL> select INITCAP('the lion') from dual;


<4> LOWER('문자열') //영문 소문자로 변환

SQL> select LOWER('MY NAME IS KHS') from dual;

UPPER('문자열') //영문 대문자로 변환

SQL> select upper('the lion') from dual;



- <5> **LPAD**('뒤에 채울 문자열', 전체문자열길이, '남는 앞공간에 채울 문자열')
- ```
SQL> select LPAD('yangssem', 13, '*#') from dual;
```
- ```
SQL> select LPAD('₩200,000', 13, ' ') from dual;
```
- <6> **RPAD**('앞에 채울 문자열', 자리수, '남는 뒷공간에 채울 문자열')
- ```
SQL> select RPAD('yangssem', 13, '@') from dual;
```
- <7> **LTRIM**('문자열', '지우고 싶은 문자') 왼쪽지우기
- ```
SQL> select LTRIM('aa123456aa', 'aa') from dual;
```
- <8> **RTRIM**('문자열', '지우고 싶은 문자') 오른쪽 지우기
- ```
SQL> select RTRIM('aa123456aa', 'aa') from dual;
```
- <9> **REPLACE**('원문', '타켓 문자열', '수정 문자열')
- ```
SQL> select 'yang and sam' from dual;
```
- ```
SQL> select REPLACE('yang and sam', 'sa', 'sse') from dual;
```
- ```
SQL> select replace(DEPARTMENT_NAME,
```
- 'IT', 'information technology') from departments;
- <10> **SUBSTR**('문자열', 자리수, 갯수) 일부분 추출
- ```
SQL> select SUBSTR('ABCDEFGG', 2, 1) from dual;
```
- // first\_name의 두번째 자리가 'i'인 사원의 이름을 출력
- ```
SQL> select first_name from employees where first_name like '_i%';
```
- ```
SQL> select first_name from employees
```
- where SUBSTR(first\_name , 2, 1) ='i';
- 

<11> **ASCII**('문자')

SQL> select ASCII('A') from dual;

<12> **INSTR**('원문자열', '찾고싶은문자열') 첫출현 위치찾을때

SQL> select INSTR('010-2222-8888','-') from dual;

**INSTR**('원문자열', '찾고싶은 문자열', 몇번째부터, 몇번째중복문자인가)

SQL> select INSTR('CORPORATE FLOOR','OR', 1, 2) from dual;

<13> **LENGTH**('문자열')

SQL> select length('yangssem') from dual;

<14> **GREATEST**('문자열1', '문자열2','문자열3') 최대값

**GREATEST**(정수1,정수2,정수) 최대값

SQL> select GREATEST('CC', 'ABCDE', 'CA') from dual;

SQL> select GREATEST('12', '132', '34') from dual;

SQL> select GREATEST(12,132, 34) from dual;

<15> **LEAST**('문자열1', '문자열2','문자열3') 최소값

**LEAST**(정수1,정수2,정수) 최소값

SQL> select LEAST('AB', 'ABCDE', 'CA') from dual;

SQL> select LEAST('12', '3', '34') from dual;

SQL> select LEAST(12, 3, 34) from dual;

SQL> select LEAST('가나', '가다', '다라') from dual;



<16> NVL(컬럼명, 숫자) 만약null일때 대체값 설정,  
SQL> select salary + (salary \* nvl(commission\_pct, 5) ) from employees;  
NVL2(컬럼명, 숫자1, 숫자2) 만약null일때 숫자2, null이 아닐때 숫자1  
SQL> select salary + (salary \* nvl2(commission\_pct, 5, 10))  
from employees;

<17> DECODE(A, B, 참 리턴값)  
select last\_name, job\_id, salary,  
decode(job\_id, 'IT\_PROG', salary\*1.1) AS "실수령액"  
from employees;  
DECODE(A,B,참 리턴값, 거짓 리턴값)  
select last\_name, job\_id, salary,  
decode(job\_id, 'IT\_PROG', salary\*1.1,salary) AS "실수령액"  
from employees;

DECODE(A, B1,참 리턴값, B2,참 리턴값, B3,참 리턴값,...,  
해당사항없을때 리턴값)  
select last\_name, job\_id, salary,  
decode(job\_id, 'IT\_PROG', salary\*1.1,  
'AD\_PRES', salary\*1.2,  
'AD\_VP',salary\*1.3, salary) AS "실수령액"  
from employees;



```
<18> CASE A WHEN B1 THEN 참 리턴값
 WHEN B2 THEN 참 리턴값
 ELSE 상기조건에 해당 안되는 경우 리턴값
END [별칭]
 select last_name, job_id, salary,
 case job_id when 'IT_PROG' then salary*1.1
 when 'AD_PRES' then salary*1.2
 when 'AD_VP' then salary*1.3
 else salary
 end AS "실수령액"
 from employees;
```



## 2) 날짜 함수

### <1> SYSDATE

SQL> select SYSDATE from dual;

### <2> ADD\_MONTHS(날짜컬럼 or 날짜데이터, 숫자) 개월을 더하거나 뺄때

SQL> select HIRE\_DATE, ADD\_MONTHS(HIRE\_DATE, 3)  
from employees where EMPLOYEE\_ID = 100;

SQL> select HIRE\_DATE, ADD\_MONTHS(HIRE\_DATE, -3)  
from employees where EMPLOYEE\_ID = 100;

SQL> select ADD\_MONTHS('13/04/20', 12) from dual;  
select ADD\_MONTHS(SYSDATE , 12) from dual;

### <3> LAST\_DAY(날짜컬럼 or 날짜데이터) 해당 월의 마지막 일(30,31,28,29)

SQL> select HIRE\_DATE, LAST\_DAY(HIRE\_DATE) from employees;

SQL> select LAST\_DAY('13/04/20') from dual;

### <4> NEW\_TIME(날짜컬럼 or 날짜데이터, 'GMT', 'PDT')

gmt(그리니치 표준시)를 pdt(태평양 연안 표준시)로 변환

SQL> select HIRE\_DATE, NEW\_TIME(HIRE\_DATE, 'GMT', 'PDT')  
from employees where EMPLOYEE\_ID=100;





- <5> **MONTHS\_BETWEEN**(날짜컬럼or날짜데이터1, 날짜컬럼or날짜데이터2)  
SQL> select hire\_date, sysdate, **MONTHS\_BETWEEN**(sysdate, hire\_date)  
from employees where employee\_id = 100;  
// 결과의 단위는 '달(월)'
- <6> **NEXT\_DAY**(날짜컬럼or날짜데이터, 요일숫자) //1:일요일,2:월요일,3:화요일...  
SQL> select NEXT\_DAY(sysdate ,3) from dual;  
//당일 이후 가장가까운 화요일  
SQL> select first\_name, hire\_date, NEXT\_DAY(hire\_date, 2)  
from employees;  
//입사일 이후 가장가까운 월요일



### 3) 문자 변환 함수

-> **TO\_CHAR**(날짜컬럼or날짜데이터, '??')

<1> 'D' 숫자로 반환 (1:일요일,2:월요일,3:화요일...7:토요일)

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'D') from dual;

<2> 'DAY' 텍스트반환 >> 수요일

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'DAY') from dual;

<3> 'DY' 텍스트 한글자 반환 >> 수

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'DY') from dual;

<4> 'DD' 일자 정수 >> 20

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'DD') from dual;

<5> 'MM' 달 정수 >> 03

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'MM') from dual;

<6> 'MONTH' 달 텍스트 >> 3월

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'MONTH') from dual;

<7> 'YY' 연도 정수 >> 13

SQL> select SYSDATE, TO\_CHAR(SYSDATE, 'YY') from dual;



<8> 'YYYY'

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'YYYY') from dual;
```

```
SQL> select hire_date, TO_CHAR(hire_date, 'YYYY') from employees;
```

<9> 'DD-MM-YY'(원하는 순서로 조합 가능)

```
SQL> select hire_date, TO_CHAR(hire_date, 'DD-MM-YY')
 from employees;
```

<10> 'fmDD-MM-YY' : 예(13-03-09)일경우 13-3-9와같이 0 생략

```
SQL> select hire_date, TO_CHAR(hire_date, 'fmDD-MM-YY')
 from employees;
```

<11> 'YYYY-MM-DD'

```
SQL> select hire_date, TO_CHAR(hire_date, 'YYYY-MM-DD')
 from employees;
```

<12> 'fmYYYY-MM-DD'

```
SQL> select hire_date, TO_CHAR(hire_date, 'fmYYYY-MM-DD')
 from employees;
```



<13> 'HH' or 'HH12'

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'HH') from dual;
```

<14> 'HH24'

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'HH24') from dual;
```

<15> 'MI' 분

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'MI') from dual;
```

<16> 'SS' 초

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'SS') from dual;
```

<17> 'AM' or 'PM'

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'PM HH') from dual;
```

```
SQL> select SYSDATE, TO_CHAR(SYSDATE, 'AM HH:MI:SS') from dual;
```

<18> 'YYYY-MM-DD AM HH:MI:SS DAY' 텍스트위치는 이동가능

```
SQL> select TO_CHAR(SYSDATE, 'YYYY-MM-DD DAY AM HH:MI:SS')
as 현재시간 from dual;
```



#### 4) 숫자 변환 함수

-> **TO\_NUMBER('문자로된 정수또는 실수')**

SQL> SELECT TO\_NUMBER('100')+1 FROM dual;

SQL> SELECT TO\_NUMBER('3.14') FROM dual;

#### 5) 날짜 변환 함수

-> **TO\_DATE('날짜 형태의 문자열',**

**'날짜변환포맷(앞 문자열을 해석하는 순서)')**

SQL> select TO\_DATE(sysdate, 'yy/mm/dd') FROM dual;

SQL> select TO\_DATE('13-03-20', 'dd/mm/yy') FROM dual;

#### 6) 시스템 함수

-> **USER(field의 데이터 사용자를 리턴)**

SQL> select USER from dual;



## 7) 숫자 함수

<1> **ABS**(숫자) // 절대값 즉 - 또는 + 부호를 떼어낸 수

SQL> select ABS(-30) from dual;

<2> **CEIL**(숫자) // 올림값

SQL> select CEIL(11.001) from dual;

<3> **FLOOR**(숫자) // 내림값

SQL> select FLOOR(4.999) from dual;

<4> **ROUND**(숫자) // 소수점 1째에서 반올림함.

SQL> select ROUND(22.5) from dual;

**ROUND**(숫자, 자릿수) // 자릿수가 반올림되어짐.

SQL> select ROUND(22.567, 2) from dual;

<5> **COS**(숫자[rad])

SQL> select cos(180\*3.14/180) from dual;

<6> **SIN**(숫자[rad])

SQL> select SIN(180\*3.14/180) from dual;

<7> **TAN**(숫자[rad])

SQL> select TAN(45\*3.14/180) from dual;

<8> **MOD**(숫자1, 숫자2) 숫자2로 나눈 나머지

SQL> select MOD(11, 4) from dual;

<9> **POWER**(숫자1, 숫자2)

SQL> select POWER(2, 7) from dual; 2에7제곱

select POWER(2, 8) from dual; 2에8제곱

<10> **TRUNC**(숫자1, 숫자2) 숫자2자릿수 빼고 버림

SQL> select TRUNC(99.123456, 3) from dual;



(5) 복수행(그룹) 함수(\*\*\*\*\*)

1) COUNT(컬럼명)

SQL> select COUNT(first\_name) from employees;

SQL> select COUNT(employee\_id) from employees;

SQL> select COUNT(\*) from employees;

SQL> select COUNT(commission\_pct) from employees; //null 제외됨

2) SUM(컬럼명)

SQL> select SUM(salary) from employees;

SQL> select SUM(commission\_pct) from employees;

3) AVG(컬럼명)

SQL> select AVG(commission\_pct) from employees;

//NULL갯수 제외

SQL> select AVG(NVL(commission\_pct,0)) from employees;

//NULL값을 0으로 대체되기때문에 갯수 포함

4) MAX(컬럼명) 와 MIN(컬럼명)

SQL> select MAX(salary) from employees;

SQL> select MIN(salary) from employees;



## GROUP BY 절과 HAVING 절 ## 중요(\*\*\*\*\*)

group by : 그룹화된 결과를 얻어냄.

조건절을 쓸때는 where 절은 쓸 수 없고 having

// 각 부서별 연봉의 평균을 구해라.

```
SQL> select AVG(salary), department_id from employees
 GROUP BY department_id ;
```

```
SQL> select ROUND(AVG(salary)), department_id from employees
 GROUP BY department_id;
```

```
SQL> select ROUND(AVG(salary)), department_id from employees
 GROUP BY department_id
 HAVING department_id is not null;
```





//Q1:연봉 8000이상인 직원들의 부서별 평균연봉의 반올림값을 출력하라.

```
SELECT DEPARTMENT_ID,ROUND(AVG(SALARY)) FROM EMPLOYEES
WHERE SALARY>=8000
GROUP BY DEPARTMENT_ID;
```

주의 ) 부서별 평균연봉이8000 이상인 직원들의 평균연봉을 반올림해서 출력하는 것과는 다르다.

```
SELECT DEPARTMENT_ID,ROUND(AVG(SALARY)) FROM EMPLOYEES
GROUP BY DEPARTMENT_ID
having avg(salary)>=8000;
```

//Q2:연봉 8000이상인 직원들의 부서별 평균연봉의 반올림값  
을 부서번호의 내림차순으로 정렬하라

```
SELECT DEPARTMENT_ID,ROUND(AVG(SALARY)) FROM EMPLOYEES
WHERE SALARY>=8000
GROUP BY DEPARTMENT_ID
ORDER BY DEPARTMENT_ID DESC;
```

//Q3:연봉 8000이상인 직원들의 부서별 평균연봉의 반올림값  
을 평균연봉의 반올림값으로 오름차순 정렬하라

```
SELECT DEPARTMENT_ID,ROUND(AVG(SALARY)) AS R FROM EMPLOYEES
WHERE SALARY>=8000
GROUP BY DEPARTMENT_ID
ORDER BY R ASC;
```



//Q4:연봉 10000이상인 직원들의 부서별 평균연봉의 반올림값  
을 부서번호로 오름차순 정렬하라

```
SELECT DEPARTMENT_ID,ROUND(AVG(SALARY)) AS R FROM
EMPLOYEES
WHERE SALARY>=10000
GROUP BY DEPARTMENT_ID
ORDER BY DEPARTMENT_ID ASC;
```

//Q5:각 부서별 같은 업무를 하는 사람의 인원수를 구하여  
부서번호,업무명,인원수를 출력하라.  
(단, 부서번호와 업무명으로 각각 내림차순 정렬!)

```
SELECT DEPARTMENT_ID,JOB_ID,count(*) FROM EMPLOYEES
GROUP BY DEPARTMENT_ID,JOB_ID
ORDER BY DEPARTMENT_ID DESC,JOB_ID DESC;
```



//Q6:사원 테이블에서 연봉과 사원ID를 출력하되  
연봉이 12000인 사원은 고액연봉자라고 출력하라.

```
select salary,employee_id,decode(salary,12000,'고액연봉자')
from employees;
```

//Q7:사원 테이블에서 연봉과 사원ID를 출력하되  
연봉이 12000이상인 사원은 비고에 고액연봉자라고  
출력하라. (단, 최고연봉액은 24000이다.)

```
select salary,employee_id,case when salary between 12000 and
24000 then '고액연봉자' end 비고 from employees;
```

//Q8:사원 테이블에서 연봉과 사원ID를 출력하되  
연봉이 20000이상인 사원은 1급연봉자,  
연봉이 15000 에서 19000 이하인 사원은 2급연봉자,  
연봉이 기타연봉인 사원은 3급연봉자 라고 출력하라.  
(단, 최고연봉액은 24000이다.)

```
select salary,employee_id,
case when salary between 12000 and 24000 then '1급연봉자'
when salary between 12000 and 19000 then '2급연봉자'
else '3급'
end 비고
from employees;
```

