

# 4장. 오라클 함수

오라클 DB – SQL 디벨로퍼

ORACLE®  
DATABASE 11g



# 단일행 함수

## 문자 타입 함수

| 함수      | 설명                    | 예                       | 결과     |
|---------|-----------------------|-------------------------|--------|
| LOWER   | 값을 소문자로 변환            | LOWER('ABCD')           | abcd   |
| UPPER   | 값을 대문자로 변환            | UPPER('abcd')           | ABCD   |
| INITCAP | 첫번째 글자만 대문자로 변환       | INITCAP ('abcd')        | Abcd   |
| SUBSTR  | 문자열중 일부분을 선택          | SUBSTR('ABC', 1, 2)     | AB     |
| REPLACE | 특정 문자열을 찾아 바꾼다        | REPLACE('AB', 'A', 'E') | EB     |
| CONCAT  | 두 문자열을 연결(   연산자와 같다) | CONCAT('A', 'B')        | AB     |
| LENGTH  | 문자열의 길이를 구한다.         | LENGTH('AB')            | 2      |
| INSTR   | 명명된 문자의 위치를 구한다.      | INSTR('ABCD', 'D')      | 4      |
| LPAD    | 왼쪽부터 특정문자로 자리를 채움     | LPAD('ABCD', 6, '*')    | **ABCD |
| RPAD    | 오른쪽부터 특정문자로 자리를 채움    | RPAD('ABCD', 6, '*')    | ABCD** |



# 단일행 함수

## 문자 타입 함수

```
SELECT last_name,  
       LOWER(last_name) LOWER적용,  
       UPPER(last_name) UPPER적용,  
       email,  
       INITCAP(email) INITCAP적용  
FROM t_employee2;
```

|   | LAST_NAME | LOWER적용 | UPPER적용 | EMAIL    | INITCAP적용 |
|---|-----------|---------|---------|----------|-----------|
| 1 | King      | king    | KING    | SKING    | Sking     |
| 2 | Kochhar   | kochhar | KOCHHAR | NKOCHHAR | Nkochhar  |
| 3 | De Haan   | de haan | DE HAAN | LDEHAAN  | Ldehaan   |
| 4 | Hunold    | hunold  | HUNOLD  | AHUNOLD  | Ahunold   |
| 5 | Ernst     | ernst   | ERNST   | BERNST   | Bernst    |

```
-- job_id의 첫째 자리부터 시작해서 두 개의 문자 출력  
SELECT job_id, SUBSTR(job_id, 1, 2) 직무코드  
FROM t_employee2;
```

|   | JOB_ID  | 직무코드 |
|---|---------|------|
| 1 | AD PRES | AD   |
| 2 | AD VP   | AD   |
| 3 | AD VP   | AD   |
| 4 | IT PROG | IT   |
| 5 | IT PROG | IT   |



# 단일행 함수

## 문자 타입 함수

```
-- job_id 문자열 값이 ACCOUNT이면 ACCNT로 출력  
SELECT job_id, REPLACE(job_id, 'ACCOUNT', 'ACCNT') 결과  
FROM t_employee2;
```

|    |    |         |    |       |
|----|----|---------|----|-------|
| 8  | IT | PROG    | IT | PROG  |
| 9  | FI | MGR     | FI | MGR   |
| 10 | FI | ACCOUNT | FI | ACCNT |
| 11 | FI | ACCOUNT | FI | ACCNT |
| 12 | FI | ACCOUNT | FI | ACCNT |

```
--first_name에 대해 12자리의 문자열 자리를 *를 채워서 출력하기  
SELECT first_name, LPAD(first_name, 12, '*') 결과  
FROM t_employee2;
```

| ⚡ FIRST_NAME | ⚡ 결과         |
|--------------|--------------|
| 1 Steven     | *****Steven  |
| 2 Neena      | *****Neena   |
| 3 Lex        | *****Lex     |
| 4 Alexander  | ***Alexander |
| 5 Bruce      | *****Bruce   |



# 단일행 함수

## 숫자 타입 함수

| 함수    | 설명             | 예                | 결과   |
|-------|----------------|------------------|------|
| ROUND | 숫자를 반올림한다.     | ROUND(12.583, 1) | 12.6 |
| TRUNC | 숫자를 절삭한다.(버림)  | TRUNC(12.583, 1) | 12.5 |
| MOD   | 나누기 후 나머지를 구한다 | MOD(15, 2)       | 1    |
| CEIL  | 숫자를 정수로 올림한다.  | CEIL(15.351)     | 16   |
| FLOOR | 숫자를 정수로 내림한다.  | FLOOR(15.351)    | 15   |
| POWER | 거듭제곱을 구한다.     | POWER(2, 3)      | 8    |
| SQRT  | 제곱근을 구한다.      | SQRT(4)          | 2    |



# 단일행 함수

-- salary를 30일로 나눈 후 소수 자리수에 따라 반올림한 값 출력

```
SELECT salary,  
       salary/30 일급,  
       ROUND(salary/30, 1) 결과1,  
       ROUND(salary/30, 0) 결과2,  
       ROUND(salary/30, -1) 결과3
```

```
FROM t employee2;
```

|   | SALARY | 일급                                   | 결과1   | 결과2 | 결과3 |
|---|--------|--------------------------------------|-------|-----|-----|
| 1 | 24000  |                                      | 800   | 800 | 800 |
| 2 | 17000  | 566.66666666666666666666666666666667 | 566.7 | 567 | 570 |
| 3 | 17000  | 566.66666666666666666666666666666667 | 566.7 | 567 | 570 |
| 4 | 9000   |                                      | 300   | 300 | 300 |
| 5 | 6000   |                                      | 200   | 200 | 200 |

-- salary를 30일로 나눈 후 소수 자리수에 따라 절삭(버린)한 값 출력

```
SELECT salary,  
       salary/30 일급,  
       TRUNC(salary/30, 1) 결과1,  
       TRUNC(salary/30, 0) 결과2,  
       TRUNC(salary/30, -1) 결과3
```

```
FROM t_employee2;
```

[illegible]

# 단일행 함수

## 날짜 연산 규칙

| 함수            | 설명            | 반환값  |
|---------------|---------------|------|
| Date + Number | 날짜에서 일수를 더한다. | Date |
| Date - Number | 날짜에서 일수를 뺀다.  | Date |
| Date - Date   | 날짜에서 날짜를 뺀다.  | 일수   |

## 날짜 함수

| 함수            | 설명                       | 예                                 |
|---------------|--------------------------|-----------------------------------|
| MONTH_BETWEEN | 두 날짜 사이의 월수를 계산          | MONTH_BETWEEN(SYSDATE, HIRE_DATE) |
| ADD_MONTHS    | 월을 날짜에 더한다.              | ADD_MONTHS(HIRE_DATE, 5)          |
| NEXT_DAY      | 명시된 날짜부터 돌아오는 요일의 날짜를 출력 | NEXT_DAY(HIRE_DATE, 1)            |



# 단일행 함수

```
-- department_id가 100인 직원에 대해 입사후 총 개월수 출력
-- department_id, 오늘 날짜, hire_date, 총 개월수 출력
SELECT department_id, SYSDATE, hire_date,
       TRUNC(MONTHS_BETWEEN(SYSDATE, hire_date), 0) 총_개월수
FROM t_employee2
WHERE department_id = 100;
```

|   | DEPARTMENT_ID | SYSDATE    | HIRE_DATE  | 총_개월수 |
|---|---------------|------------|------------|-------|
| 1 | 100           | 2021/03/01 | 2002/08/17 | 222   |
| 2 | 100           | 2021/03/01 | 2002/08/16 | 222   |
| 3 | 100           | 2021/03/01 | 2005/09/28 | 185   |
| 4 | 100           | 2021/03/01 | 2005/09/30 | 185   |
| 5 | 100           | 2021/03/01 | 2006/03/07 | 179   |
| 6 | 100           | 2021/03/01 | 2007/12/07 | 158   |

```
-- employee_id가 100과 106사이인 직원의 hire_date에 3개월을 더한 값과
-- 3개월을 뺀 값 출력
SELECT employee_id, hire_date,
       ADD_MONTHS(hire_date, 3) 더하기_결과,
       ADD_MONTHS(hire_date, -3) 빼기_결과
FROM t_employee2
WHERE employee_id BETWEEN 100 AND 106;
```

|   | EMPLOYEE_ID | HIRE_DATE  | 더하기_결과     | 빼기_결과      |
|---|-------------|------------|------------|------------|
| 1 | 100         | 2003/06/17 | 2003/09/17 | 2003/03/17 |
| 2 | 101         | 2005/09/21 | 2005/12/21 | 2005/06/21 |
| 3 | 102         | 2001/01/13 | 2001/04/13 | 2000/10/13 |
| 4 | 103         | 2006/01/03 | 2006/04/03 | 2005/10/03 |
| 5 | 104         | 2007/05/21 | 2007/08/21 | 2007/02/21 |
| 6 | 105         | 2005/06/25 | 2005/09/25 | 2005/03/25 |
| 7 | 106         | 2006/02/05 | 2006/05/05 | 2005/11/05 |



# 단일행 함수

## 변환 함수

### 자동 데이터 타입 변환

| FROM             | TO           |
|------------------|--------------|
| VARCHAR2 또는 CHAR | NUMBER(숫자)   |
| VARCHAR2 또는 CHAR | DATE(날짜)     |
| NUMBER           | VARCHAR2(문자) |
| DATE             | VARCHAR2(문자) |

```
-- 자동 타입 변환
SELECT 1 + '2'
FROM DUAL;
```

|   |       |   |
|---|-------|---|
| 1 | 1+'2' | 3 |
|---|-------|---|

DUAL 테이블은 오라클에서  
제공되는 테이블로 함수 결  
과값을 출력할 때 사용

```
-- dual 테이블
SELECT * FROM dual;

SELECT SYSDATE FROM DUAL;
```



# 단일행 함수

## 변환 함수

### 수동 데이터 타입 변환

| FROM      | TO                              |
|-----------|---------------------------------|
| TO_CHAR   | 숫자, 문자, 날짜 값을 형식을 VARCHAR2로 변환  |
| TO_NUMBER | 문자를 숫자 타입으로 변환                  |
| TO_DATE   | 날짜를 나타내는 문자열을 지정 형식의 날짜 타입으로 변환 |



# 단일행 함수

-- 날짜 형식 변환

```
SELECT TO_CHAR(SYSDATE, 'YY') 년도,  
       TO_CHAR(SYSDATE, 'YYYY') 년도_4,  
       TO_CHAR(SYSDATE, 'MM') 월,  
       TO_CHAR(SYSDATE, 'DD') 일,  
       TO_CHAR(SYSDATE, 'YY/MM/DD') 날짜  
FROM DUAL;
```

|   | 년도 | 년도_4 | 월  | 일  | 날짜       |
|---|----|------|----|----|----------|
| 1 | 21 | 2021 | 03 | 01 | 21/03/01 |

-- 시간 형식 변환

```
SELECT TO_CHAR(SYSDATE, 'HH:MI:SS') 시간형식,  
       TO_CHAR(SYSDATE, 'YYYY/MM/DD HH:MI:SS PM') 날짜와시간  
FROM DUAL;
```

|   | 시간형식     | 날짜와시간                  |
|---|----------|------------------------|
| 1 | 05:16:40 | 2021/03/01 05:16:40 오전 |

-- 숫자 형식 변환

```
SELECT TO_NUMBER('123')  
FROM DUAL;
```

|   | TO_NUMBER('123') |
|---|------------------|
| 1 | 123              |

-- 날짜 형식 변환

```
SELECT TO_DATE('20190515', 'YYMMDD')  
FROM DUAL;
```

|   | TO_DATE('20190515', 'YYMMDD') |
|---|-------------------------------|
| 1 | 2019/05/15                    |



# 그룹 함수

**그룹 함수** – 단일 행 함수와 달리 여러 행에 대해 함수가 적용되어 하나의 결과를 나타내는 함수

| 함수    | 예             |
|-------|---------------|
| COUNT | COUNT(salary) |
| SUM   | SUM(salary)   |
| AVG   | AVG(salary)   |

```
SELECT COUNT(*) AS 급여행수
FROM t_employee2;

SELECT COUNT(salary) AS 급여행수
FROM t_employee2;
```

|   | 급여행수 |
|---|------|
| 1 | 107  |



# 그룹 함수

**그룹 함수** – 단일 행 함수와 달리 여러 행에 대해 함수가 적용되어 하나의 결과를 나타내는 함수

```
SELECT SUM(salary) 합계, ROUND(AVG(salary),2) 평균  
FROM employees;
```

```
SELECT MAX(salary) 최대값, MIN(salary) 최소값  
FROM employees;
```

```
SELECT MAX(first_name) 최대문자값, MIN(first_name) 최소문자값  
FROM employees;
```

|   | 합계     | 평균      |
|---|--------|---------|
| 1 | 691416 | 6461.83 |

|   | 최대값   | 최소값  |
|---|-------|------|
| 1 | 24000 | 2100 |

|   | 최대문자값   | 최소문자값 |
|---|---------|-------|
| 1 | Winston | Adam  |



# GROUP BY – HAVING

## GROUP BY : 그룹으로 묶기 & HAVING 절 사용

특정 열의 데이터 값을 기준으로 그룹화하여 다른 열에 그룹 함수를 적용해야 할 때 사용, 조건을 사용할 땐 **HAVING** 사용

```
SELECT job_id 직무, SUM(salary) 직무별_총급여, AVG(salary) 직무별_평균급여
FROM employees
WHERE employee_id >= 100
GROUP BY job_id
ORDER BY 직무별_평균급여 DESC;
```

| 직무        | 직무별_총급여 | 직무별_평균급여 |
|-----------|---------|----------|
| 1 AD PRES | 24000   | 24000    |
| 2 AD VP   | 34000   | 17000    |
| 3 MK MAN  | 13000   | 13000    |
| 4 SA MAN  | 61000   | 12200    |
| 5 AC MGR  | 12008   | 12008    |
| 6 FI MGR  | 12008   | 12008    |
| 7 PU MAN  | 11000   | 11000    |

```
SELECT job_id 직무, SUM(salary) 직무별_총급여, AVG(salary) 직무별_평균급여
FROM t_employee2
WHERE employee_id >= 100
GROUP BY job_id
HAVING SUM(salary) > 30000
ORDER BY 직무별_총급여 DESC;
```

| 직무           | 직무별_총급여 | 직무별_평균급여 |
|--------------|---------|----------|
| 1 SA REP     | 250500  | 8350     |
| 2 SH CLERK   | 64300   | 3215     |
| 3 SA MAN     | 61000   | 12200    |
| 4 ST CLERK   | 55700   | 2785     |
| 5 FI ACCOUNT | 39600   | 7920     |
| 6 ST MAN     | 36400   | 7280     |
| 7 AD VP      | 34000   | 17000    |

# 그룹 함수 – RANK()

## RANK() 함수 – 데이터 값에 순위 정하기

| 함수         | 설명                                    | 순위 예            |
|------------|---------------------------------------|-----------------|
| RANK       | 공통 순위를 출력하되 공통 순위만큼 건너뛰어 다음 순위를 출력한다. | 1, 2, 2, 4, ... |
| DENSE_RANK | 공통 순위를 출력하되 공통 건너뛰지 않고 다음 순위를 출력한다.   | 1, 2, 2, 3, ... |



# 그룹 함수 – RANK()

## RANK() 함수 – 데이터 값에 순위 정하기

### RANK() OVER(ORDER BY 열 이름)

```
-- RANK() 순위
SELECT employee_id, last_name,
       salary,
       RANK() OVER(ORDER BY salary DESC) 급여_RANK,
       DENSE_RANK() OVER(ORDER BY salary DESC) 급여_DENSE_RANK
FROM employees;
```

| EMPLOYEE_ID | LAST_NAME | SALARY | 급여_RANK | 급여_DENSE_RANK |
|-------------|-----------|--------|---------|---------------|
| 100         | King      | 24000  | 1       | 1             |
| 101         | Kochhar   | 17000  | 2       | 2             |
| 102         | De Haan   | 17000  | 2       | 2             |
| 145         | Russell   | 14000  | 4       | 3             |
| 146         | Partners  | 13500  | 5       | 4             |
| 201         | Hartstein | 13000  | 6       | 5             |
| 108         | Greenberg | 12008  | 7       | 6             |
| 205         | Higgins   | 12008  | 7       | 6             |
| 147         | Errazuriz | 12000  | 9       | 7             |
| 168         | Ozer      | 11500  | 10      | 8             |



# 그룹 함수 – RANK()

RANK() 함수 – 그룹으로 묶어서 순위를 정해야 할때

**RANK() OVER(PARTITION BY 열 이름 ORDER BY 열 이름)**

```
-- 같은 부서 내에서 직원의 급여 순위 정하기
SELECT department_id, first_name,
       salary,
       RANK() OVER(PARTITION BY department_id ORDER BY salary DESC) 급여_RANK,
       DENSE_RANK() OVER(PARTITION BY department_id ORDER BY salary DESC) 급여_DENSE_RANK
FROM employees;
```

| DEPARTMENT_ID | FIRST_NAME | SALARY | 급여_RANK | 급여_DENSE_RANK |
|---------------|------------|--------|---------|---------------|
| 10            | Jennifer   | 4400   | 1       | 1             |
| 20            | Michael    | 13000  | 1       | 1             |
| 20            | Pat        | 6000   | 2       | 2             |
| 30            | Den        | 11000  | 1       | 1             |
| 30            | Alexander  | 3100   | 2       | 2             |
| 30            | Shelli     | 2900   | 3       | 3             |

|    |          |      |    |    |
|----|----------|------|----|----|
| 50 | Kelly    | 3800 | 10 | 10 |
| 50 | Jennifer | 3600 | 11 | 11 |
| 50 | Renske   | 3600 | 11 | 11 |
| 50 | Trenna   | 3500 | 13 | 12 |
| 50 | Julia    | 3400 | 14 | 13 |
| 50 | Jason    | 3300 | 15 | 14 |
| 50 | Laura    | 3300 | 15 | 14 |
| 50 | Winston  | 3200 | 17 | 15 |



# 일반 함수 – NVL() 함수

## NVL 함수 – NULL 값 처리하기

특정 열의 행에 대한 데이터 값이 없다면 데이터 값은 null이 된다. 테이블을 정의할 때 NOT NULL을 지정하면 null 값을 가질 수 없다.

### NVL (열이름, 치환값)

```
-- NVL (열이름, 치환값) --
```

```
SELECT *  
FROM employees  
WHERE manager_id IS NULL;
```

```
SELECT last_name, NVL(manager_id, employee_id)  
FROM employees  
WHERE manager_id IS NULL;
```

| LAST_NAME | NVL(MANAGER_ID, EMPLOYEE_ID) |
|-----------|------------------------------|
| King      | 100                          |



# 일반 함수 – NVL() 함수

## NVL 함수 – NULL 값 처리하기

```
SELECT first_name, salary * commission_pct  
FROM employees  
ORDER BY commission_pct;
```

```
SELECT first_name, salary * NVL(commission_pct, 1)  
FROM employees  
ORDER BY commission_pct;
```

|    | FIRST_NAME | SALARY*COMMISSION_PCT |
|----|------------|-----------------------|
| 31 | Louise     | 2250                  |
| 32 | Janette    | 3500                  |
| 33 | Patrick    | 3325                  |
| 34 | Allan      | 3150                  |
| 35 | John       | 5600                  |
| 36 | Steven     | (null)                |
| 37 | Neena      | (null)                |
| 38 | Lex        | (null)                |
| 39 | Alexander  | (null)                |
| 40 | Bruce      | (null)                |
| 41 | David      | (null)                |
| 42 | Valli      | (null)                |
| 43 | Diana      | (null)                |



|    | FIRST_NAME | SALARY*NVL(COMMISSION_PCT,1) |
|----|------------|------------------------------|
| 31 | Louise     | 2250                         |
| 32 | Janette    | 3500                         |
| 33 | Patrick    | 3325                         |
| 34 | Allan      | 3150                         |
| 35 | John       | 5600                         |
| 36 | Steven     | 24000                        |
| 37 | Neena      | 17000                        |
| 38 | Lex        | 17000                        |
| 39 | Alexander  | 9000                         |
| 40 | Bruce      | 6000                         |
| 41 | David      | 4800                         |
| 42 | Valli      | 4800                         |
| 43 | Diana      | 4200                         |



# DECODE 함수() vs CASE 표현식

## DECODE 함수 – (IF~THEN~ELSE)

DECODE (열이름, 조건 값, **변경 값**, 기본값)

조건에 맞으면 변경값  
조건에 맞지않으면 기본값

| LAST_NAME   | DEPARTMENT_ID | 원래급여  | 조정급여  | 인상여부  |
|-------------|---------------|-------|-------|-------|
| 1 King      | 90            | 24000 | 24000 | 미인상   |
| 2 Kochhar   | 90            | 17000 | 17000 | 미인상   |
| 3 De Haan   | 90            | 17000 | 17000 | 미인상   |
| 4 HUNOLD    | 60            | 9000  | 9900  | 10%인상 |
| 5 Ernst     | 60            | 6000  | 6600  | 10%인상 |
| 6 Austin    | 60            | 4800  | 5280  | 10%인상 |
| 7 Pataballa | 60            | 4800  | 5280  | 10%인상 |
| 8 Lorentz   | 60            | 4200  | 4620  | 10%인상 |
| 9 Greenberg | 100           | 12008 | 12008 | 미인상   |
| 10 FAVIET   | 100           | 9000  | 9000  | 미인상   |
| 11 Chen     | 100           | 8200  | 8200  | 미인상   |
| 12 Sciarra  | 100           | 7700  | 7700  | 미인상   |
| 13 Urman    | 100           | 7800  | 7800  | 미인상   |

```
-- DECODE 함수 --
SELECT last_name,
       department_id,
       salary 원래급여,
       DECODE(department_id, 60, salary*1.1, salary) 조정급여,
       DECODE(department_id, 60, '10%인상', '미인상') 인상여부
FROM employees;
```

# DECODE() 함수 vs CASE 표현식

## CASE WHEN 표현식

```
CASE  
  
    WHEN 조건 1 THEN 출력 값1  
  
    ELSE 출력값 2  
  
END
```

```
-- CASE ~ WHEN 절 --  
SELECT last_name, department_id, salary 원래급여,  
       CASE  
           WHEN department_id = 60 THEN salary*1.1  
           ELSE salary  
       END AS 조정급여,  
       CASE  
           WHEN department_id = 60 THEN '10% 인상'  
           ELSE '미인상'  
       END AS 인상여부  
FROM employees;
```

| LAST_NAME   | DEPARTMENT_ID | 원래급여  | 조정급여  | 인상여부  |
|-------------|---------------|-------|-------|-------|
| 1 King      | 90            | 24000 | 24000 | 미인상   |
| 2 Kochhar   | 90            | 17000 | 17000 | 미인상   |
| 3 De Haan   | 90            | 17000 | 17000 | 미인상   |
| 4 Hunold    | 60            | 9000  | 9900  | 10%인상 |
| 5 Ernst     | 60            | 6000  | 6600  | 10%인상 |
| 6 Austin    | 60            | 4800  | 5280  | 10%인상 |
| 7 Pataballa | 60            | 4800  | 5280  | 10%인상 |
| 8 Lorentz   | 60            | 4200  | 4620  | 10%인상 |
| 9 Greenberg | 100           | 12008 | 12008 | 미인상   |
| 10 Faviet   | 100           | 9000  | 9000  | 미인상   |
| 11 Chen     | 100           | 8200  | 8200  | 미인상   |