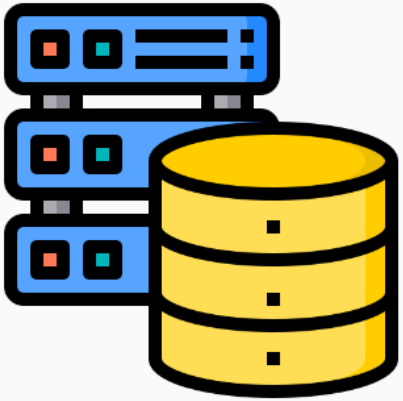
4장. 함수



SQL 내장 함수

SQL 내장함수

상수나 속성 이름을 입력 값으로 받아 단일 값을 결과로 반환한다.

모든 내장 함수는 최초에 선언될 때 유효한 입력값을 받아야 한다.

예를 들어 수학 함수의 입력값은 정수 또는 실수 여야 한다.

SELECT 절과 WHERE 절, UPDATE 절 등에서 모두 사용 가능하다.

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
ABS	절대값을 구한다	ABS(-10)	10
POWER	거듭제곱을 구한다.	POWER(2, 3)	8
SQRT	제곱근을 구한다.	SQRT(4)	2

숫자 함수

```
-- 숫자 함수
-- FROM 절이 없는 SELECT문 : 오라클은 가상 테이블인 dual을 사용함
-- 절대값 구하기
SELECT  ABS(-10) FROM dual;

-- 3.875를 소수 첫째자리까지 반올림한 값을 구하시오
SELECT  ROUND(3.875, 1) FROM dual;
```

ABS(-10)	ABS(10)
10	10

ROUND(3.875,1)
3.9

숫자 함수

```
-- 고객별 평균 주문 금액을 백원 단위로 반올림한 값을 구하시오
SELECT custid AS 고객번호,
       ROUND(AVG(saleprice), -2) AS 평균금액
FROM orders
GROUP BY custid;
```

고객번호	평균금액
1	13000
2	7500
3	10300
4	16500

```
SELECT custid 고객번호, COUNT(*) 주문수, SUM(saleprice) 총액
FROM orders
GROUP BY custid;

SELECT custid 고객번호, ROUND(SUM(saleprice)/COUNT(*), -2) 평균금액
FROM orders
GROUP BY custid;
```

숫자 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 emp;
```

	SALARY	일급	결과1	결과2	결과3
1	3000000	100000	100000	100000	100000
2	2600000	86666.666666...	86666.7	86667	86670

```
-- salary를 30일로 나눈 후 소수 자리수에 따라 절삭(버림)한 출력
SELECT salary,
       salary/30 일급,
       TRUNC(salary/30, 1) 결과1,
       TRUNC(salary/30, 0) 결과2,
       TRUNC(salary/30, -1) 결과3
FROM emp;
```

	SALARY	일급	결과1	결과2	결과3
1	3000000	100000	100000	100000	100000
2	2600000	86666.6666666...	86666.6	86666	86660

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**

문자타입 함수

문자 타입 함수

```sql
-- 소문자로 변경하기
SELECT LOWER('ABCD') RESULT FROM DUAL;

-- SUBSTR(문자, 인덱스, 글자수) : 글자수 추출하기
SELECT SUBSTR('ABC', 1, 2) RESULT FROM DUAL;

-- REPLACE(문자, 이전문자, 새로운문자) : 문자 변경하기
SELECT REPLACE('ABC', 'A', 'E') RESULT FROM DUAL;

-- CONCAT(문자1, 문자2) - 문자 연결
SELECT CONCAT('A', 'B') RESULT FROM DUAL;

-- 연결연산자 - '||'
SELECT '안녕' || '하세요' RESULT FROM DUAL;
```

```sql
-- LPAD(문자, 문자수, 기호) - 기호를 왼쪽부터 채움
-- RPAD(문자, 문자수, 기호) - 기호를 오른쪽부터 채움
SELECT LPAD('cloud', 10, '*') RESULT FROM DUAL;
SELECT RPAD('cloud', 10, '*') RESULT FROM DUAL;
```

LPAD('CLOUD',10,'*')
*****cloud

문자타입 함수

```sql
-- 굿스포츠에서 출판한 도서의 제목과 제목의 문자 수, 바이트 수를 검색
-- 한글 3Byte, 영어, 특수기호 - 1Byte
SELECT bookname,
       LENGTH(bookname)문자수,
       LENGTHB(bookname) 바이트수
FROM book
WHERE publisher = '굿스포츠';
```

	BOOKNAME	문자수	바이트수
1	축구의 역사	6	16
2	피겨 교본	5	13
3	양궁의 실제	6	16

```sql
-- 도서 제목에 야구가 포함된 도서를 농구로 변경하여 검색
SELECT bookid,
       REPLACE(bookname, '축구', '농구') bookname,
       publisher
FROM book;
```

BOOKID	BOOKNAME	PUBLISHER
1	농구의 역사	굿스포츠
2	농구아는 여자	나무수
3	농구의 이해	대한미디어
4	골프 바이블	대한미디어
5	피겨 교본	굿스포츠

```sql
-- 고객 이름에서 같은 성을 가진 사람의 인원수를 구하시오
-- GROUP BY절에는 함수도 포함할 수 있음
SELECT SUBSTR(name, 1, 1) 성,
       COUNT(*) 인원
FROM customer
GROUP BY SUBSTR(name, 1, 1);
```

성	인원
박	2
김	1
안	1
류	1
손	1

실습 문제

➢ 부서 테이블을 생성하고 자료를 추가한 후 아래의 결과대로 출력하시오

```sql
-- 부서 테이블(dept)
CREATE TABLE dept(
    deptid     NUMBER PRIMARY KEY,    -- 기본키
    deptname   VARCHAR2(20) NOT NULL, -- NULL 불허
    location   VARCHAR2(20) NOT NULL
);

-- 부서 자료 추가 --
INSERT INTO dept(deptid, deptname, location)
VALUES (10, '전산팀', '서울');
INSERT INTO dept(deptid, deptname, location)
VALUES (20, '관리팀', '인천');
INSERT INTO dept(deptid, deptname, location)
VALUES (30, '마케팅팀', '수원');
```

	팀명
1	전산
2	관리
3	마케팅

날짜 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 (이후날짜, 이전날짜)	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)

날짜 함수

```
-- 날짜 출력
SELECT SYSDATE FROM DUAL;

-- 날짜와 시간
SELECT SYSTIMESTAMP FROM DUAL;

-- 20일전의 날짜 출력
SELECT SYSDATE - 20 FROM DUAL;

-- 3개월 후의 날짜 출력
SELECT ADD_MONTHS(SYSDATE, 3) 결과
FROM DUAL;
-- 3개월 전의 날짜 출력
SELECT ADD_MONTHS(SYSDATE, -3) 결과
FROM DUAL;
```

날짜 함수

```
-- 특정일에서 3개월 전의 날짜 출력
SELECT ADD_MONTHS('2023/04/01', -3) 결과
FROM DUAL;
```

	결과
1	23/01/01

```
-- 특정한 날에서 10일후(특정한 날: 문자형 -> 날짜형)
-- SELECT TO_DATE('2023/10/01') + 10 FROM DUAL;
SELECT TO_DATE('2023-10-01') + 10 결과
FROM DUAL;
```

	결과
1	23/10/11

```
-- 입사일 : 2022-1-1, 퇴사일 : 2023-1-31(월수 계산)
SELECT
    ROUND(MONTHS_BETWEEN(TO_DATE('2022-12-31'),
                TO_DATE('2022-1-1')), 0) 총개월수
FROM DUAL;
```

날짜 함수

```
-- 서점은 주문일로부터 10일후 매출을 확정한다.
-- 각 주문의 확정일자를 구하시오.
SELECT orderid 주문번호,
       orderdate 주문일,
       TO_DATE(orderdate) + 10 확정일
FROM orders;
```

주문번호	주문일	확정
1	18/07/01	18/07/11
2	18/07/03	18/07/13
3	18/07/03	18/07/13
4	18/07/04	18/07/14
5	18/07/05	18/07/15
6	18/07/07	18/07/17
7	18/07/07	18/07/17
8	18/07/08	18/07/18
9	18/07/09	18/07/19
10	18/07/10	18/07/20

날짜 함수

```sql
-- 주문번호가 6에서 10사이인 도서의 주문일에 3개월을 더한값을 구하시오.
-- 1. 주문번호가 6~10인 도서 검색
-- 2. 주문일에 3개월 더하기, 빼기
SELECT orderid 주문번호,
       ADD_MONTHS(orderdate, 3) 더하기결과,
       ADD_MONTHS(orderdate, -3) 빼기결과
FROM orders
--WHERE orderid >=6 AND orderid <= 10;
WHERE orderid BETWEEN 6 AND 10;
```

주문번호	주문일	더하기_결과	빼기_결과
6	18/07/07	18/10/07	18/04/07
7	18/07/07	18/10/07	18/04/07
8	18/07/08	18/10/08	18/04/08
9	18/07/09	18/10/09	18/04/09
10	18/07/10	18/10/10	18/04/10

```sql
-- 주문번호가 10인 도서의 주문일로부터 오늘까지의 총 개월수를 구하시오
SELECT orderid 주문번호, orderdate 주문일, SYSDATE 오늘,
       TRUNC(MONTHS_BETWEEN(SYSDATE, orderdate), 0) 총개월수
FROM orders
WHERE orderid = 10;
```

주문번호	주문일	오늘	총개월수
10	18/07/10	22/07/14	48

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+'2'
1	3

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

```
-- 숫자 형식 변환
SELECT TO_NUMBER('123')
FROM DUAL;
```

	TO_NUMBER('123')
1	123

```
-- 날짜 형식(지정된 날짜 형식으로 출력됨)
SELECT TO_DATE('2022-06-30') FROM DUAL;
```

TO_DATE('2022-06-30','YYYY-MM-DD')
22/06/30

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
22	2022	07	14	22/07/14

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

시간	날짜와시간
05:45:20	2022/07/14 05:45:20

DECODE() 함수 vs CASE 표현식

CASE WHEN 표현식

```
CASE
        WHEN 조건1 THEN 결과1
        WHEN 조건2 THEN 결과2
        ELSE 결과 3
END  칼럼명
```

DECODE 함수 – (IF~THEN~ELSE)

```
DECODE (열이름, 조건 값, 변경 값, 기본값)
```

DECODE 함수() vs CASE 표현식

```sql
-- DECODE(칼럼명, 조건, 참, 거짓) 함수 -IF 함수와 유사함
-- 조건에 특정값이 오고, 범위는 올 수 없음
-- Male, Female
SELECT ename,
       DECODE(gender, '남자', 'M', 'F') gender,
       salary
FROM emp;
```

```sql
SELECT ename,
       CASE
         WHEN gender = '남자' THEN 'M'
         ELSE 'F'
       END gender,
       salary
FROM emp;
```

	ENAME	GENDER	SALARY
1	이강	M	3000000
2	김산	F	2600000
3	박신입	F	(null)

DECODE 함수() vs CASE 표현식

```sql
-- 급여가 350만원 이상이면 직급을 '과장'로 표시하고,
-- 250만원 이상이면 '대리'이고 아니면 '사원'으로 표시
SELECT ename,
       salary,
       CASE
         WHEN salary >= 3500000  THEN '과장'
         WHEN salary >= 2500000  THEN '대리'
         ELSE '사원'
       END 급여기준
FROM emp;
```

	ENAME	SALARY	급여기준
1	이강	3000000	대리
2	김산	2600000	대리
3	박신입	(null)	사원
4	오상식	5000000	과장

DECODE 함수() vs CASE 표현식

	DATA	DATA2
1	-1	-1
2	0	0
3	-1	-1
4	0	0
5	-1	-1

```
CREATE TABLE TEST(
    COL1 NUMBER(1)
);

INSERT INTO TEST VALUES (NULL);
INSERT INTO TEST VALUES (0);
INSERT INTO TEST VALUES (NULL);
INSERT INTO TEST VALUES (0);
INSERT INTO TEST VALUES (NULL);

SELECT
    CASE WHEN T.COL1 IS NULL THEN -1
         ELSE 0
    END AS DATA,
    DECODE(T.COL1, NULL, -1, T.COL1) AS DATA2
FROM TEST T;
```

일반 함수 – NVL() 함수

NVL 함수 – NULL 값 처리하기

NULL값이란 아직 지정되지 않은 값을 말한다. 지정되지 않았다는 것은 값을 알수도 없고 적용할 수도 없다는 뜻이다.

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

NVL (인수1, 인수2)

인수1의 값이 NULL이 아닌 경우 인수1을 반환하고

인수 1의 값이 NULL일 경우 인수2를 반환함

일반 함수 – NVL() 함수

NVL 함수 – NULL 값 처리하기

```
-- NVL(인수1, 인수2)
-- 인수1이 NULL이 아니면 인수1 출력, NULL이면 인수2 출력
SELECT ename,
       NVL(salary, 0) salary
FROM emp;
```

	ENAME	SALARY
1	이강	3000000
2	김산	2600000
3	박신입	0
4	오상식	5000000

일반 함수 – NVL() 함수

NVL 함수 – NULL 값 처리하기

```
CREATE TABLE K1(
    ID VARCHAR2(3),
    CNT NUMBER(2)
);

INSERT INTO K1 VALUES ('가', 5);
INSERT INTO K1 VALUES ('나', NULL);
INSERT INTO K1 VALUES ('다', 5);
INSERT INTO K1 VALUES ('라', NULL);
INSERT INTO K1 VALUES ('마', 10);
```

일반 함수 – NVL() 함수

NVL 함수 – NULL 값 처리하기

```
SELECT ID, CNT
FROM K1;

SELECT NVL(CNT, 0)
FROM K1;

SELECT COUNT(NVL(CNT, 0)) COUNT FROM K1; --5

SELECT SUM(NVL(CNT, 0))/4 SUM FROM K1;    --5

SELECT AVG(NVL(CNT, 0)) AVERAGE FROM K1; --4
-- NULL을 5로 변경
SELECT NVL(CNT, 5)
FROM K1;

SELECT MIN(NVL(CNT, 5)) AVERAGE FROM K1; --5
```

	ID	CNT
1	가	5
2	나	(null)
3	다	5
4	라	(null)
5	마	10

	NVL(CNT,0)
1	5
2	0
3	5
4	0
5	10

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 건너 뛰지 않고 다음 순위를 출력한다.	1, 2, 2, 3, …

그룹 함수 – RANK()

RANK() 함수 – 데이터 값에 순위 정하기

RANK() OVER(ORDER BY 열 이름)

DENSE_RANK() OVER(ORDER BY 열 이름)

```
INSERT INTO emp VALUES (100, '이강', '남자', 3000000, '2019-01-01');
INSERT INTO emp VALUES (101, '김산', '여자', 2600000, '2020-05-15');
INSERT INTO emp VALUES (102, '오상식', '남자', 5000000, '2015-02-22');
INSERT INTO emp VALUES (103, '박신입', '여자', '',  '2023-10-01');
INSERT INTO emp VALUES (105, '우영우', '여자', 2600000, '2021-10-13');
INSERT INTO emp VALUES (103, '이신입', '남자', '2000000',  '2022-10-01');
```

그룹 함수 – RANK()

RANK() 함수 – 데이터 값에 순위 정하기

```
--RANK() 함수
SELECT ename,
       salary,
       RANK() OVER(ORDER BY salary DESC) 급여_RANK,
       DENSE_RANK() OVER(ORDER BY salary DESC) 급여_DENSE_RANK
FROM emp;
```

	ENAME	SALARY	급여_RANK	급여_DENSE_RANK
1	오상식	5000000	1	1
2	이강	3000000	2	2
3	김산	2600000	3	3
4	우영우	2600000	3	3
5	이신입	2000000	5	4

sum over() – 누적 합계

SUM(칼럼명) OVER(ORDER BY 칼럼명)

```
CREATE TABLE dept(
    -- 칼럼이름 자료형
    deptno  VARCHAR2(4) PRIMARY KEY,
    deptname  VARCHAR2(20) NOT NULL,
    office  VARCHAR2(10)
);

INSERT INTO dept(deptno, deptname, office) VALUES ('1000', '인사팀', '서울');
INSERT INTO dept(deptno, deptname, office) VALUES ('1001', '전산팀', '수원');
INSERT INTO dept VALUES ('1002', '전산팀', '수원');
INSERT INTO dept(deptno, deptname) VALUES ('1003', '영업팀');
```

sum over() – 누적 합계

SUM(칼럼명) OVER(ORDER BY 칼럼명)

```
-- 사원 테이블(부서와 관계를 맺은)
CREATE TABLE employee(
    empno    NUMBER(3) PRIMARY KEY,      -- 사원번호
    ename    VARCHAR2(20) NOT NULL,      -- 사원이름
    sal      NUMBER(10),                 -- 급여
    createdate  DATE DEFAULT SYSDATE,    -- 등록일
    gender   VARCHAR2(10),               -- 성별
    deptno   VARCHAR2(4),                -- 부서번호
    -- 외래키(FOREIGN KEY) 제약조건
    CONSTRAINT emp_fk FOREIGN KEY(deptno)
    REFERENCES dept(deptno)
    -- ON DELETE CASCADE(부서를 삭제하면 참조하고 있는 사원도 삭제)
);
```

sum over() – 누적 합계

SUM(칼럼명) OVER(ORDER BY 칼럼명)

```
INSERT INTO employee VALUES (100, '이강', 2500000, SYSDATE, '남자', '1000');
INSERT INTO employee VALUES (101, '이산', 2000000, SYSDATE, '여자', '1001');
INSERT INTO employee VALUES (102, '박달', 1500000, SYSDATE, '남자', '1002');
INSERT INTO employee VALUES (103, '강하늘', 3500000, SYSDATE, '', '1003');
INSERT INTO employee VALUES (104, '양우주', 4500000, SYSDATE, '여자', '1000');
INSERT INTO employee VALUES (105, '강남', 2600000, SYSDATE, '남자', '1000');
INSERT INTO employee VALUES (106, '이해', 2500000, SYSDATE, '남자', '1000');
```

sum over() – 누적 합계

```
-- GROUP BY 절(소속등 그룹화) 조건절 - HAVING 사용
-- 부서별 급여 총액을 구하시오
SELECT deptno 부서, SUM(sal) 급여총액, AVG(sal) 급여평균
FROM employee
GROUP BY deptno
HAVING AVG(sal) >= 3000000
ORDER BY AVG(sal) DESC;
```

```
-- 부서별 급여 총액(부서 이름 출력)
SELECT a.deptno,
       a.deptname,
       SUM(b.sal)
FROM dept a, employee b
WHERE a.deptno = b.deptno
GROUP BY a.deptno, a.deptname;
```

	DEPTNO	DEPTNAME	SUM(B.SAL)
1	1000	인사팀	12100000
2	1001	전산팀	2000000
3	1002	전산팀	1500000
4	1003	영업팀	3500000

sum over()

SUM(칼럼명) OVER(ORDER BY 칼럼명)

```
-- 부서별 급여 총액이 500만원 이상인 자료 검색
SELECT deptno, SUM(sal)
FROM employee
GROUP BY deptno
HAVING SUM(sal) > 5000000;

-- 급여 누적값 계산
-- SUM(칼럼명) OVER(ORDER BY 칼럼명)
SELECT empno,
       ename,
       sal,
       SUM(sal) OVER(ORDER BY empno) "sal_sum" -- 오름차순 누적
FROM employee;
```

	EMPNO	ENAME	SAL	sal_sum
1	100	이강	2500000	2500000
2	101	이산	2000000	4500000
3	102	박달	1500000	6000000
4	103	강하늘	3500000	9500000
5	104	양우주	4500000	14000000
6	105	강남	2600000	16600000
7	106	이해	2500000	19100000

LOLLUP() 함수 vs CUBE()

LOLLUP(칼럼명, 칼럼명)

GROUP BY의 칼럼에 대해서 소계를 만들어 준다.
GROUP BY 구문에 칼럼이 두 개 이상 오면 순서에 따라 결과가 달라진다.

CUBE(칼럼명, 칼럼명)

GROUP BY의 칼럼에 대해서 결합 가능한 모든 집계를 계산한다.
다차원 집계를 제공하여 다양하게 데이터를 분석할 수 있다

LOLLUP() 함수 vs CUBE()

✓ **DEPT 테이블 생성**

```
-- LOLLUP, CUBE
CREATE TABLE DEPT(
    DEPT_NO VARCHAR2(3),
    JOB_NM  VARCHAR2(50),
    SALARY  NUMBER(5)
);

INSERT INTO DEPT VALUES ('100', '증권사', 3300);
INSERT INTO DEPT VALUES ('100', '관리자', 4300);
INSERT INTO DEPT VALUES ('200', '증권사', 5000);
INSERT INTO DEPT VALUES ('200', '데이터분석가', 4000);
INSERT INTO DEPT VALUES ('200', '관리자', 6000);
```

LOLLUP() 함수 vs CUBE()

✓ 그룹함수 – Group By 절

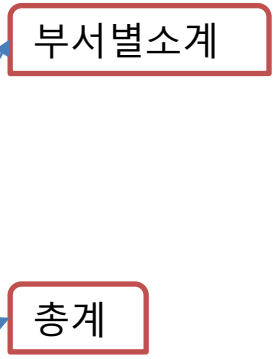
```
-- 부서 전체의 인원수와 급여 합계
SELECT COUNT(*) 인원수, SUM(salary) 급여합계
FROM dept;

-- 부서별, 직업 이름별 인원수, 급여합계
SELECT dept_no, job_nm, SUM(salary) 급여합계
FROM dept
GROUP BY dept_no, job_nm;
```

LOLLUP() 함수 vs CUBE()

✓ **LOLLUP(칼럼명, 칼럼명)**

```
-- LOLLUP()
SELECT DEPT_NO, JOB_NM, SUM(SALARY)
FROM DEPT
GROUP BY ROLLUP(DEPT_NO, JOB_NM)
ORDER BY DEPT_NO;
```

	DEPT_NO	JOB_NM	SUM(SALARY)
1	100	관리자	4300
2	100	증권사	3300
3	100	(null)	7600
4	200	관리자	6000
5	200	데이터분석가	4000
6	200	증권사	5000
7	200	(null)	15000
8	(null)	(null)	22600

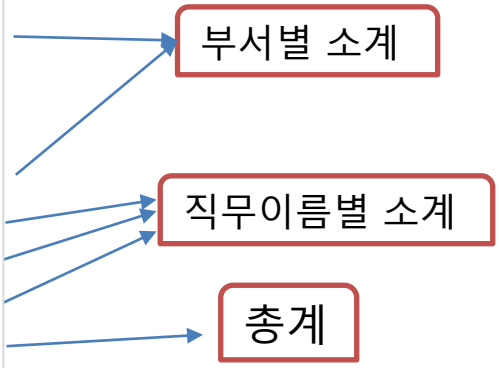


LOLLUP() 함수 vs CUBE()

✓ **CUBE(칼럼명, 칼럼명)**

```
-- CUBE()
SELECT DEPT_NO, JOB_NM, SUM(SALARY)
FROM DEPT
GROUP BY CUBE(DEPT_NO, JOB_NM)
ORDER BY DEPT_NO;
```

	DEP...	JOB_NM	SUM(SALARY)
1	100	관리자	4300
2	100	증권사	3300
3	100	(null)	7600
4	200	관리자	6000
5	200	데이터분석가	4000
6	200	증권사	5000
7	200	(null)	15000
8	(null)	관리자	10300
9	(null)	데이터분석가	4000
10	(null)	증권사	8300
11	(null)	(null)	22600

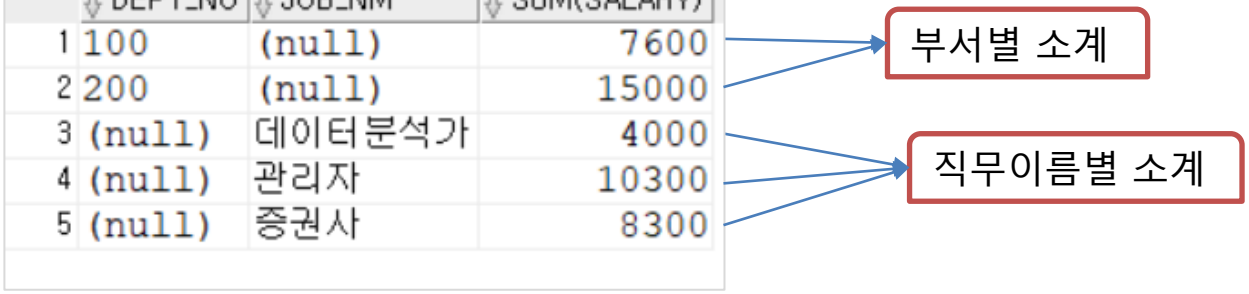


LOLLUP() 함수 vs CUBE()

✓ **GROUPING SETS(칼럼명, 칼럼명)**

```
-- GROUPING SETS()
SELECT DEPT_NO, JOB_NM, SUM(SALARY)
FROM DEPT
GROUP BY GROUPING SETS(DEPT_NO, JOB_NM)
ORDER BY DEPT_NO;
```

	DEPT_NO	JOB_NM	SUM(SALARY)
1	100	(null)	7600
2	200	(null)	15000
3	(null)	데이터분석가	4000
4	(null)	관리자	10300
5	(null)	증권사	8300



LOLLUP() 함수 vs CUBE()

✓ **GROUPING(칼럼명)**

```
-- GROUPING() 함수는 ROLLUP(), CUBE() 등과 함께 사용
-- GROUPING() 된 결과는 1로 출력됨
SELECT GROUPING(DEPT_NO), GROUPING(JOB_NM)
       DEPT_NO, JOB_NM, SUM(SALARY)
FROM DEPT
GROUP BY ROLLUP(DEPT_NO, JOB_NM)
ORDER BY DEPT_NO;
```

GROUPING(DEPT_NO)	DEPT_NO	JOB_NM	SUM(SALARY)
0	0	증권사	3300
0	0	관리자	4300
0	0	증권사	5000
0	0	데이터분석가	4000
0	0	관리자	6000
0	1	(null)	7600
0	1	(null)	15000
1	1	(null)	22600

LOLLUP() 함수 vs CUBE()

✓ **GROUPING(칼럼명)**

```
-- NULL을 소계, 합계로 변경하기
SELECT
        CASE GROUPING(DEPT_NO)
            WHEN 1 THEN '합계'
            ELSE DEPT_NO
        END DEPT_NO,
        CASE GROUPING(JOB_NM)
            WHEN 1 THEN '소계'
            ELSE JOB_NM
        END JOB_NM,
        SUM(SALARY)
FROM DEPT
GROUP BY ROLLUP(DEPT_NO, JOB_NM)
ORDER BY DEPT_NO;
```

	DEPT_NO	JOB_NM	SUM(SALARY)
1	100	관리자	4300
2	100	소계	7600
3	100	증권사	3300
4	200	관리자	6000
5	200	증권사	5000
6	200	데이터분석가	4000
7	200	소계	15000
8	합계	소계	22600

등급 매기기

- NTILE() 함수

  주어진 수만큼 행들을 n등분한 후 현재 행에 해당하는 등급을 구하는 함수

```
CREATE TABLE exam_score (
    이름     VARCHAR(20),
    국어     NUMBER,
    영어     NUMBER,
    수학     NUMBER
);
```

```
INSERT INTO EXAM_SCORE VALUES ('고하나', 116, 77, 75);
INSERT INTO EXAM_SCORE VALUES ('이하나', 101, 69, 80);
INSERT INTO EXAM_SCORE VALUES ('박하나', 118, 62, 60);
INSERT INTO EXAM_SCORE VALUES ('정하나', 96, 72, 70);
INSERT INTO EXAM_SCORE VALUES ('최하나', 103, 77, 55);
INSERT INTO EXAM_SCORE VALUES ('김하나', 78, 66, 61);
INSERT INTO EXAM_SCORE VALUES ('한하나', 85, 72, 75);
INSERT INTO EXAM_SCORE VALUES ('장하나', 99, 70, 53);
INSERT INTO EXAM_SCORE VALUES ('윤하나', 105, 75, 69);
INSERT INTO EXAM_SCORE VALUES ('임하나', 117, 68, 73);
```

등급 매기기

```
-- 과목별 등급 검색
SELECT 이름,
       NTILE(8) OVER(ORDER BY 국어 DESC) 국어등급,
       국어
  FROM exam_score;

-- '최하나'가 없는 9명인 경우
SELECT 이름,
       NTILE(8) OVER(ORDER BY 국어 DESC) 국어등급,
       국어
  FROM exam_score
 WHERE 이름 <> '최하나';
```

	이름	국어등급	국어
1	박하나	1	118
2	임하나	1	117
3	고하나	2	116
4	윤하나	2	105
5	최하나	3	103
6	이하나	4	101
7	장하나	5	99
8	정하나	6	96
9	한하나	7	85
10	김하나	8	78

	이름	국어등급	국어
1	박하나	1	118
2	임하나	1	117
3	고하나	2	116
4	윤하나	3	105
5	이하나	4	101
6	장하나	5	99
7	정하나	6	96
8	한하나	7	85
9	김하나	8	78

등급 매기기

```
-- 전체 과목(언어영역 120, 영어, 수학 - 80)
SELECT 이름,
       NTILE(8) OVER(ORDER BY 국어 DESC) 국어등급,
       국어,
       NTILE(8) OVER(ORDER BY 영어 DESC) 영어등급,
       영어,
       NTILE(8) OVER(ORDER BY 수학 DESC) 수학등급,
       수학
  FROM exam_score;
```

	이름	국어등급	국어	영어등급	영어	수학등급	수학
1	박하나	1	118	8	62	6	60
2	임하나	1	117	6	68	2	73
3	고하나	2	116	1	77	1	75
4	윤하나	2	105	2	75	4	69
5	최하나	3	103	1	77	7	55
6	이하나	4	101	5	69	1	80
7	장하나	5	99	4	70	8	53
8	정하나	6	96	2	72	3	70
9	한하나	7	85	3	72	2	75
10	김하나	8	78	7	66	5	61

누적값 구하기

누적값

- SUM(칼럼명) OVER(ORDER BY 칼럼명) AS SUM

부분 누적값

- SUM(칼럼명) OVER(PARTIAN BY 칼럼명 ORDER BY 칼럼명) AS SUM

누적값 구하기

```
CREATE TABLE EMPLOYEES (
    DEPARTMENT_ID   VARCHAR(2),
    NAME            VARCHAR(15),
    HIRE_DATE       DATE,
    SALARY          NUMBER
);

INSERT INTO EMPLOYEES VALUES('10','Jennifer',TO_DATE('2013-09-17'),4400);
INSERT INTO EMPLOYEES VALUES('10','Pat',TO_DATE('2015-08-17'),6000);
INSERT INTO EMPLOYEES VALUES('10','Alexander',TO_DATE('2016-01-03'),3100);
INSERT INTO EMPLOYEES VALUES('20','Michael',TO_DATE('2014-02-17'),13000);
INSERT INTO EMPLOYEES VALUES('20','Kevin',TO_DATE('2016-01-24'),6000);
INSERT INTO EMPLOYEES VALUES('30','Den',TO_DATE('2012-12-07'),11000);
INSERT INTO EMPLOYEES VALUES('30','Shelli',TO_DATE('2015-07-24'),2900);
INSERT INTO EMPLOYEES VALUES('30','Sigal',TO_DATE('2015-12-24'),2800);
```

누적값 구하기

	DEPARTME...	NAME	SALARY	HIRE_DATE	SAL_SUM
1	10	Jennifer	4400	13/09/17	4400
2	10	Pat	6000	15/08/17	10400
3	10	Alexander	3100	16/01/03	13500
4	20	Michael	13000	14/02/17	13000
5	20	Kevin	6000	16/01/24	19000
6	30	Den	11000	12/12/07	11000
7	30	Shelli	2900	15/07/24	13900
8	30	Sigal	2800	15/12/24	16700

행 순서 관련 함수

◆ 행 순서 관련 함수

상위 행의 값을 하위에 출력하거나 하위 행의 값을 상위 행에 출력할 수 있다.

키워드	설 명
LEAD()	하위 행의 값을 상위 행에 출력함
LAG()	상위 행의 값을 하위에 출력함

행 순서 관련 함수

```
CREATE TABLE EMP_INFO (
    EMP_NO      VARCHAR(3),
    NAME        VARCHAR(20),
    SAL         NUMBER
);

INSERT INTO EMP_INFO VALUES('100','Steven',24000);
INSERT INTO EMP_INFO VALUES('101','Neena',17000);
INSERT INTO EMP_INFO VALUES('102','Lex',17000);
INSERT INTO EMP_INFO VALUES('108','Nancy',12000);
INSERT INTO EMP_INFO VALUES('109','Daniel',9000);
INSERT INTO EMP_INFO VALUES('110','John',8200);
INSERT INTO EMP_INFO VALUES('111','Ismael',7800);
INSERT INTO EMP_INFO VALUES('112','Jose Manuel',7700);
INSERT INTO EMP_INFO VALUES('113','Luis',6900);
INSERT INTO EMP_INFO VALUES('119','Karen',2500);
```

행 순서 관련 함수

```sql
-- LEAD(칼럼명, 수) : 수만큼 위로 끌고 올라옴
-- LEAD(칼럼명, 수, 0) : NULL인 경우 0으로 지정함
SELECT EMP_NO, NAME, SAL,
       LEAD(SAL, 3) OVER(ORDER BY SAL DESC) AS SAL2,
       LEAD(SAL, 3, 0) OVER(ORDER BY SAL DESC) AS SAL3
    FROM EMP_INFO;
```

EMP_NO	NAME	SAL	SAL2	SAL3
100	Steven	24000	12000	12000
101	Neena	17000	9000	9000
102	Lex	17000	8200	8200
108	Nancy	12000	7800	7800
109	Daniel	9000	7700	7700
110	John	8200	6900	6900
111	Ismael	7800	2500	2500
112	Jose Manuel	7700	(null)	0
113	Luis	6900	(null)	0
119	Karen	2500	(null)	0

행 순서 관련 함수

```
-- LAG(칼럼명, 수) : 수만큼 아래로 내려감
-- LAG(칼럼명, 수, 0) : NULL인 경우 0으로 지정함
SELECT EMP_NO, NAME, SAL,
       LAG(SAL, 3) OVER(ORDER BY SAL DESC) AS SAL2,
       LAG(SAL, 3, 0) OVER(ORDER BY SAL DESC) AS SAL3
    FROM EMP_INFO;
```

EMP_NO	NAME	SAL	SAL2	SAL3
100	Steven	24000	(null)	0
101	Neena	17000	(null)	0
102	Lex	17000	(null)	0
108	Nancy	12000	24000	24000
109	Daniel	9000	17000	17000
110	John	8200	17000	17000
111	Ismael	7800	12000	12000
112	Jose Manuel	7700	9000	9000
113	Luis	6900	8200	8200
119	Karen	2500	7800	7800