5장. 데이터 집계

IF문을 SQL로 구현하기 ①

```
1   SELECT ename, deptno, DECODE(deptno, 10, 300, 20, 400, 0) as 보너스
2     FROM emp;
```

- 위의 예제는 이름과 부서 번호와 보너스를 출력하는 쿼리입니다.
- 보너스는 DECODE 함수를 이용해서 생성한 컬럼입니다.
- DECODE(deptno,10,300,20,400,0)는 부서 번호가 10번이면 300을, 부서 번호가 20번이면 400을, 나머지 부서 번호는 0을 출력합니다.
- 맨 끝에 0은 default 값으로 앞의 값에 만족하지 않은 데이터이면 즉, 부서 번호가 10번, 20번이 아니면 출력되는 값입니다.
- IF문으로 표현하면 다음과 같습니다.

IF문	설명
IF DEPTNO = 10 THEN 300	부서 번호가 10번이면 300 출력
ELSE IF DETNO = 20 THEN 400	그렇지 않고 부서 번호가 20번이면 400 출력
ELSE 0	위의 조건에 다 해당되지 않는다면 0 출력

IF문을 SQL로 구현하기 ①

- 다음의 쿼리는 사원 번호와 사원 번호가 짝수인지 홀수 인지를 출력하는 쿼리입니다.

```
1  SELECT empno, mod(empno, 2), DECODE(mod(empno, 2), 0, '짝수', 1, '홀수') as 보너스
2    FROM emp;
```

- 위의 쿼리는 default 값이 없습니다.
- 짝수와 홀수가 아닌 숫자는 없기 때문에 따로 default 값을 지정하지 않았습니다.
- default 값은 생략 가능합니다.

IF문	설명
IF mod(empno,2) = 0 THEN '짝수'	mod(empno,2)가 0이면 '짝수, 출력
ELSE IF mod(empno,2) =1 THEN '홀수'	그렇지 않고 mod(empno,2)가 1이면 '홀수, 출력
ELSE 0	위의 조건에 다 해당되지 않는다면 0 출력

IF문을 SQL로 구현하기 ①

- 다음의 쿼리는 이름과 직업과 보너스를 출력하는데 직업이 SALESMAN이면 보너스 5000을 출력하고 나머지 직업은 보너스 2000을 출력하는 예제입니다.

```
1   SELECT ename, job, DECODE(job, 'SALESMAN', 5000, 2000) as 보너스
2     FROM emp;
```

- 이번에는 else if 조건이 없고 if 다음에 바로 else가 실행되었습니다.

IF문	설명
IF JOB = 'SALESMAN' THEN 5000	직업이 SALESMAN이면 5000 출력
ELSE 2000	그렇지 않다면 2000 출력

IF문을 SQL로 구현하기 ②

```
1  SELECT ename, job, sal, CASE WHEN sal >= 3000 THEN 500
2                               WHEN sal >= 2000 THEN 300
3                               WHEN sal >= 1000 THEN 200
4                               ELSE 0 END AS BONUS
5    FROM emp
6    WHERE job IN('SALESMAN', 'ANALYST');
```

- **위의 예제는 이름, 직업, 월급, 보너스를 출력하는 쿼리입니다.**
- **보너스는 월급이 3000 이상이면 500을 출력, 월급이 2000 이상이고 3000보다 작으면 300을 출력, 월급이 1000 이상이고 2000보다 작으면 200을 출력, 나머지 사원들은 0을 출력합니다.**

IF문	설명
IF sal>= 3000 THEN 500	월급이 3000 이상이면 500 출력
ELSE IF sal>= 2000 THEN 300	그렇지 않고 월급이 2000 이상이면 300 출력
ELSE IF sal>= 1000 THEN 200	그렇지 않고 월급이 1000 이상이면 200 출력
ELSE 0	위의 조건들이 다 아니라면 0 출력

IF문을 SQL로 구현하기 ②

- CASE문이 DECODE와 다른 점은 DECODE는 등호(=) 비교만 가능하지만, CASE는 등호(=) 비교와 부등호(〉=, <= >,〈) 둘 다 가능합니다.
- 다음의 예제는 이름, 직업, 커미션, 보너스를 출력합니다.
- 보너스는 커미션이 NULL이면 500을 출력하고 NULL이 아니면 0을 출력합니다.

```
1  SELECT ename, job, comm, CASE WHEN comm is null THEN 500
2                                ELSE 0 END BONUS
3    FROM emp
4    WHERE job IN('SALESMAN', 'ANALYST');
```

IF문	설명
IF comm is null THEN 500	커미션이 null이면 500 출력
ELSE 0	그렇지 않다면 0을 출력

IF문을 SQL로 구현하기 ②

- 다음의 쿼리는 보너스를 출력할 때 직업이 SALESMAN, ANALYST이면 500을 출력하고, 직업이 CLERK, MANAGER이면 400을 출력하고, 나머지 직업은 0을 출력하는 쿼리입니다.

```
1  SELECT ename, job, CASE WHEN job in('SALESMAN', 'ANALYST') THEN 500
2                          WHEN job in('CLERK', 'MANAGER') THEN 400
3                          ELSE 0 END as 보너스
4    FROM emp;
```

IF문	설명
IF job in ('SALESMAN' ; ANALYST') THEN 500	직업이 SALESMAN, ANALYST면 500 출력
ELSE IF job in ('CLERK'.'MANAGER') THEN 400	직업이 CLERK, MANAGER면 400 출력
ELSE 0	위의 조건이 아니라면 0 출력

최대값 출력하기

- 사원 테이블에서 최대 월급을 출력해 보겠습니다.

```
1   SELECT MAX(sal)
2     FROM emp;
```

- MAX 함수를 이용하면 최대값을 출력할 수 있습니다.
- 이번에는 직업이 SALESMAN인 사원들 중 최대 월급을 출력해 보겠습니다.

```
1   SELECT MAX(sal)
2     FROM emp;
3     WHERE job = 'SALESMAN';
```

  - 3행 : WHERE절을 사용하여 직업이 SALESMAN으로 데이터를 제한합니다.

- 다음과 같이 직업이 SALESMAN인 사원들 중에서 최대 월급을 직업과 같이 출력하기 위해 위 쿼리문에 JOB 컬럼도 같이 추가해 보겠습니다.

```
1   SELECT job, MAX(sal)
2     FROM emp;
3     WHERE job = 'SALESMAN';
```

최대값 출력하기

```
SELECT job, MAX(sal)
       *
1행 오류:
ORA-00937: 단일 그룹의 그룹 함수가 아닙니다.
```

- 이렇게 에러가 발생한 이유는 job 컬럼의 값은 여러 개의 행이 출력되려고 하는데 MAX(SAL) 값은 하나가 출력되려 하기 때문입니다.
- 그래서 GROUP BY절이 필요합니다.
- GROUP BY절은 데이터를 GROUPING합니다.
- GROUP BY JOB으로 직업을 그룹핑하면 결과가 잘 출력됩니다.

```
1  SELECT job, MAX(sal)
2     FROM emp;
3     WHERE job = 'SALESMAN'
4     GROUP BY job;
```

최대값 출력하기

- 위의 SQL 실행 순서는 다음과 같습니다.

코딩 순서	SQL	실행 순서	SQL
1	SELECT job, MAX(SAL)	4	SELECT job, MAX(SAL)
2	FROM emp	1	FROM emp
3	WHERE job='SALESMAN'	2	WHERE job='SALESMAN'
4	GROUP BY job;	3	GROUP BY job;

- 다음 예제는 부서 번호와 부서 번호별 최대 월급을 출력하는 쿼리입니다.

```
1  SELECT deptno, MAX(sal)
2    FROM emp
3    GROUP BY deptno;
```

최소값 출력하기

- 직업이 SALESMAN인 사원들 중 최소 월급을 출력해 보겠습니다.

```
1  SELECT MIN(sal)
2    FROM emp
3    WHERE job = 'SALESMAN';
```

- 위 SQL의 실행 순서는 다음과 같습니다.
- FROM절에서 사원 테이블을 가져오고 WHERE절을 실행하며 직업이 SALESMAN인 사원들로 데이터를 제한합니다.
- 마지막으로 SELECT절을 실행하면서 그 중 최소 월급을 출력합니다.
- 다음은 직업과 직업별 최소 월급을 출력하는데, ORDER BY절을 사용하여 최소 월급이 높은 것부터 출력해보겠습니다.

```
1  SELECT job, MIN(sal) as 최소값
2    FROM emp
3    GROUP BY job
4    ORDER BY 최소값 desc;
```

  - 4행 : ORDER BY절은 항상 맨 마지막에 작성하고 실행 또한 맨 마지막에 실행됩니다.

최소값 출력하기

- 그룹 함수의 특징은 WHERE절의 조건이 거짓이어도 결과를 항상 출력한다는 것입니다.
- 다음의 SQL이 실행되는지 생각해보겠습니다.

```
1   SELECT MIN(sal)
2     FROM emp
3     WHERE 1 = 2;
```

- WHERE절에 1 = 2는 조건이 거짓이지만 실행됩니다.
- 결과는 NULL로 출력됩니다.
- NULL로 출력되었는지는 다음과 같이 NVL 함수를 사용해보면 알 수 있습니다.

```
1   SELECT NVL(MIN(sal), 0)
2     FROM emp
3     WHERE 1 = 2;
```

- NVL 함수에 의해 NULL이 0으로 출력되었습니다.
- 함수는 항상 결과를 리턴합니다.
- WHERE 절의 조건이 거짓이어도 no row select라고 출력되지 않고 NULL 값을 리턴합니다.

최소값 출력하기

- 다음의 예제는 직업, 직업별 최소 월급을 출력하는데, 직업에서 SAELSMAN은 제외하고 출력하고 직업별 최소 월급이 높은 것부터 출력하는 쿼리입니다.

```
1  SELECT job, MIN(sal)
2     FROM emp
3     WHERE job != 'SALEMAN'
4     GROUP BY job
5     ORDER BY MIN(sal) desc;
```

평균값 출력하기

- 사원 테이블의 평균 월급을 출력해 보겠습니다.

```
1   SELECT AVG(comm)
2     FROM emp;
```

- 위의 결과는 커미션을 다 더한 후에 4로 나눈 값입니다.

ENAME	COMM
KING	
BLAKE	
CLARK	
JONES	
MARTIN	1400
ALLEN	300
TURNER	0
JAMES	
WARD	500
FORD	

평균값 출력하기

- 그룹 함수는 NULL 값을 무시합니다.
- NULL을 제외한 나머지 4개의 데이터인 1400, 300, 0, 500을 더한 값을 4로 나누어 평균값을 출력합니다.
- 다음과 같이 NULL 값 대신 0으로 치환해주고 평균값을 출력하면 결과값이 달라집니다.

```
1   SELECT ROUND(AVG(NVL(comm, 0)))
2     FROM emp;
```

- 157로 출력되는 이유는 커미션을 다 더한 후에 14로 나누었기 때문입니다.

평균값 출력하기

- 다음 두 개의 SQL 중 성능이 더 좋은 SQL은 무엇일까요?

	NVL 함수 사용하지 않았을 때	NVL 함수 사용했을 때
SQL	SELECT SUM(comm) FROM emp;	SELECT SUM(NVL(comm,0)) FROM emp;
결과	2200	2200

- 그룹 함수는 NULL 값을 무시하므로 NVL 함수를 사용 안 한 쿼리는 커미션을 더 할 때 NULL은 제외하고 NULL이 아닌 숫자들만 합계합니다.
- 그러나 NVL 함수를 사용해서 커미션의 NULL을 0으로 변경하면 SUM 합계를 수행할 때 0이 합계 연산에 포함되어 시간이 더 걸리게 됩니다.

토탈값 출력하기

- 부서 번호와 부서 번호별 토탈 월급을 출력해 보겠습니다.

```
1  SELECT deptno, SUM(sal)
2    FROM emp
3    GROUP BY deptno;
```

- SUM 함수를 이용하여 다음의 질문에 답을 스스로 코딩해 봅니다.
- 직업과 직업별 토탈 월급을 출력하는데 직업별 토탈 월급이 높은 것부터 출력해 보겠습니다.

```
1  SELECT job, SUM(sal)
2    FROM emp
3    GROUP BY job
4    ORDER BY SUM(sal) desc;
```

- 위의 SQL의 코딩 순서와 실행 순서는 다음과 같습니다.

코딩 순서	SQL	실행 순서	SQL
1	SELECT job, SUM(SAL)	3	SELECT job, SUM(SAL)
2	FROM emp	1	FROM emp
3	GROUP BY job	2	GROUP BY job
4	ORDER BY sum(sal) DESC;	4	ORDER BY sum(sal) DESC;

토탈값 출력하기

- 직업과 직업별 토탈 월급을 출력하는데, 직업별 토탈 월급이 4000 이상인 것만 출력해 보겠습니다.

```
1   SELECT job, SUM(sal)
2     FROM emp
3     WHERE SUM(sal) >= 4000
4     GROUP BY job;
```

```
WHERE sum(sal) >= 4000
      *
3행 오류:
ORA-00934: 그룹 함수는 허가되지 않습니다.
```

- 위와 같이 WHERE절에 그룹 함수를 사용해 조건을 주면 그룹 함수는 허가되지 않는다고 하며 에러가 발생합니다.

토탈값 출력하기

- 그룹 함수로 조건을 줄 때는 WHERE절 대신 HAVING절을 사용해야 합니다.

```
1   SELECT job, SUM(sal)
2     FROM emp
3     GROUP BY job
4     HAVING SUM(sal) >= 4000;
```

- 직업과 직업별 토탈 월급을 출력하는데 직업에서 SALESMAN은 제외하고, 직업별 토탈 월급이 4000 이상인 사원들만 출력해 보겠습니다.

```
1   SELECT job, SUM(sal)
2     FROM emp
3     WHERE job != 'SALESMAN'
4     GROUP BY job
5     HAVING SUM(sal) >= 4000;
```

토탈값 출력하기

- SELECT문의 6가지 절을 다 사용해서 SQL을 작성하면 코딩 순서와 실행 순서가 다음과 같습니다.

코딩 순서	SQL	실행 순서	SQL
1	SELECT job, SUM(SAL)	5	SELECT job, SUM(SAL)
2	FROM emp	1	FROM emp
3	WHERE job !='SALESMAN'	2	WHERE job !='SALESMAN'
4	GROUP BY job	3	GROUP BY job
5	HAVING sum(sal) >= 4000	4	HAVING sum(sal) >= 4000
6	ORDER BY sum(sal) DESC;	6	ORDER BY sum(sal) DESC;

토탈값 출력하기

- 실행 순서가 위와 같으므로 다음과 같이 GROUP BY절에 컬럼 별칭을 사용하면 에러가 발생하며 실행되지 않습니다.
- FROM절을 제일 먼저 실행하고 그 다음 WHERE절, 그리고 GROUP BY절을 실행합니다.
- 그래서 EMP 테이블에서 컬럼 별칭인 '직업'을 찾을 수 없기 때문에 실행되지 않습니다.

```
1   SELECT job as 직업, SUM(sal)
2     FROM emp
3     WHERE job != 'SALESMAN'
4     GROUP BY 직업
5     HAVING SUM(sal) >= 4000;
```

```
GROUP BY 직업
         *
4행 오류:
ORA-00904: "직업": 부적합한 식별자
```

건수 출력하기

```
1  SELECT COUNT(empno)
2    FROM emp;
```

- 위의 예제는 사원 테이블 전체 사원수를 출력하는 쿼리입니다.
- COUNT 함수는 건수를 세는 함수입니다.
- 사원수를 카운트하는 방법은 다음 두 개의 SQL 중 어느 SQL로 작성해도 결과는 같습니다.

	EMPNO를 카운트 했을 때	별표(*)로 카운트 했을 때
SQL	SELECT COUNT(empno) FROM emp;	SELECT COUNT(*) FROM emp;
결과	14	14

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		81/11/17	5000		10
7698	BLAKE	MANAGER	7839	81/05/01	2850		30
7782	CLARK	MANAGER	7839	81/05/09	2450		10
7566	JONES	MANAGER	7839	81/04/01	2975		20
7654	MARTIN	SALESMAN	7698	81/09/10	1250	1400	30
7499	ALLEN	SALESMAN	7698	81/02/11	1600	300	30
7844	TURNER	SALESMAN	7698	81/08/21	1500	0	30
7900	JAMES	CLERK	7698	81/12/11	950		30
7521	WARD	SALESMAN	7698	81/02/23	1250	500	30

건수 출력하기

- COUNT(empno)는 empno만 카운트하고 COUNT(씨는 전체 행을 하나씩 카운트합니다.
- 그러나 커미션을 카운트하면 4가 출력됩니다.
- 그룹 함수는 NULL 값을 무시하기 때문입니다.

```
1   SELECT COUNT(comm)
2     FROM emp;
```

- 커미션이 NULL이 아닌 사원들 4명만 카운트하고 결과를 출력합니다.
- 그룹 함수 COUNT는 NULL 값을 COUNT하지 않습니다.
- 그러므로 그룹 함수로 SQL을 작성할 때는 NULL 값을 연산에 포함시키지 않는다는 사실을 염두하여 작성해야 합니다.
- 잘못 작성하면 다음과 같이 결과가 달라질 수 있기 때문입니다.
- 평균값을 출력할 때는 특히 NULL 값을 생각하면서 작성해야 합니다.

	NVL 함수를 사용하지 않았을 때	NVL 함수를 사용했을 때
SQL	SELECT AVG(comm) FROM emp;	SELECT ROUND(AVG(NVL(comm,0))) FROM emp;
결과	550	157

데이터 분석 함수로 순위 출력하기 ①

- 직업이 ANALYST, MANAGER인 사워들의 이름, 직업, 월급, 월급의 순위를 출력해 보겠습니다.

```
1  SELECT ename, job, sal, RANK() over (ORDER BY sal desc) 순위
2    FROM emp
3    WHERE job IN ('ANALYST', 'MANAGER');
```

- RANK( )는 순위를 출력하는 데이터 분석 함수입니다.
- RANK( ) 뒤에 OVER 다음에 나오는 괄호 안에 출력하고 싶은 데이터를 정렬하는 SQL 문장을 넣으면 그 컬럼 값에 대한 데이터의 순위가 출력됩니다.
- 위의 예제에서는 월급이 높은 사원부터 출력되게 ORDER BY절을 사용하였으므로 월급이 높은 순서부터 순위가 부여되어 출력되었습니다.
- 출력된 순위는 1등이 두 명이어서 2등이 출력되지 않고 바로 3등이 출력되었습니다.
- 바로 2등을 출력하고자 할 때는 다음 예제에서 배울 DENSE_RANK 함수를 이용하면 됩니다.

데이터 분석 함수로 순위 출력하기 ①

- 이번에는 직업별로 월급이 높은 순서대로 순위를 부여해서 각각 출력해 보겠습니다.

```
1  SELECT ename, job, sal, RANK() over (PARTITION BY job
2                                       ORDER BY sal desc) AS 순위
3    FROM emp;
```

- 직업별로 묶어서 순위를 부여하기 위해 ORDER BY 앞에 PARTITION BY JOB을 사용했습니다.

데이터 분석 함수로 순위 출력하기 ②

- **직업이 ANALYST, MANAGER인 사원들의 이름, 직업, 월급, 월급의 순위를 출력하는데 순위 1위인 사원이 두 명이 있을 경우 다음 순위가 3위로 출력되지 않고 2위로 출력해 보겠습니다.**

```
1  SELECT ename, job, sal, RANK() over (ORDER BY sal desc) AS RANK,
2                          DENSE_RANK() over (ORDER BY sal DESC) AS DENSE_RANK
3    FROM emp
4    WHERE job in ('ANALYST', 'MANAGER');
```

  - 1~2행 : RANK 함수는 순위 1 위가 두 명이어서 다음에 바로 3위를 출력했지만, DENSE_RANK는 2 위로 출력했습니다.

- **다음의 예제는 81년도에 입사한 사원들의 직업, 이름, 월급, 순위를 출력하는데, 직업별로 월급이 높은 순서대로 순위를 부여한 쿼리입니다.**

```
1  SELECT ename, job, sal, RANK() over (PARTITION BY job
2                          ORDER BY sal desc) AS 순위
3    FROM emp
4    WHERE hiredate BETWEEN TO_DATE('1981/01/01', 'RRRR/MM/DD')
5                       AND TO_DATE('1981/12/31', 'RRRR/MM/DD');
```

데이터 분석 함수로 순위 출력하기 ②

- DENSE_RANK 바로 다음에 나오는 괄호에도 다음과 같이 데이터를 넣고 사용할 수 있습니다.

```
1   SELECT DENSE_RANK(2975) within GROUP(ORDER BY sal DESC) 순위
2     FROM emp;
```

- 위의 예제는 월급이 2975인 사원은 사원 테이블에서 월급의 순위가 어떻게 되는지 출력하는 쿼리입니다.
- DENSE_RANK 바로 다음에 나오는 괄호 안에 값을 입력하고 그 값이 데이터 전체에서의 순위가 어떻게 되는지 출력하는 쿼리입니다. WITHIN은 '~ 이내'라는 뜻입니다.
- WITHIN GROUP 즉, 어느 그룹 이내에서 2975의 순위가 어떻게 되는지 보겠다는 것입니다.
- 어느 그룹이 바로 group 바로 다음에 나오는 괄호 안의 문법입니다.
- 월급이 높은 순서대로 정렬해 놓은 데이터의 그룹 안에서 2975의 순위를 출력하는 것입니다.

데이터 분석 함수로 순위 출력하기 ②

- 다음의 예제는 입사일 81년 11월 17일인 사원 테이블 전체 사원들 중 몇 번째로 입사한 것인지 출력하는 쿼리입니다.

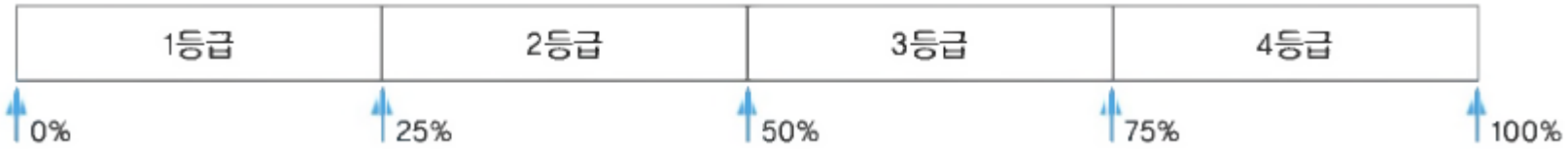
```
1  SELECT DENSE_RANK('81/11/17') within GROUP(ORDER BY hiredate ASC) 순위
2    FROM emp;
```

데이터 분석 함수로 등급 출력하기

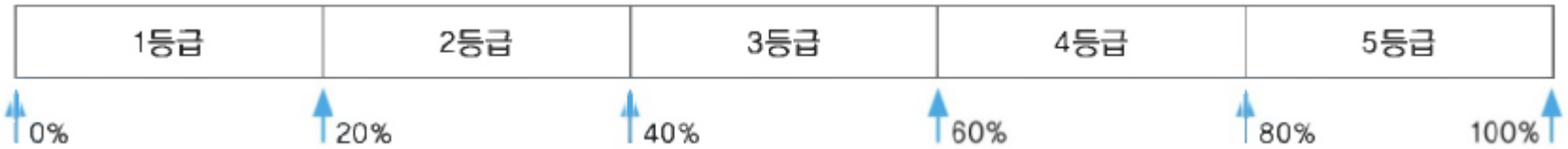
- 이름과 월급, 직업, 월급의 등급을 출력해 보겠습니다.
- 월급의 등급은 4등급으로 나눠 1등급(0～25%), 2등급(25～50%), 3등급 (50～75%), 4등급(75～100%)으로 출력해 보겠습니다.

```
1   SELECT ename, job, sal,
2          NTILE(4) over (order by sal desc nulls last) 등급
3     FROM emp
4     WHERE job in ('ANALYST','MANAGER','CLERK');
```

  - 2행 : 월급이 높은 순서대로 정렬한 결과로 4등분하여 등급을 부여합니다.



- NTILE 함수에 숫자 4를 사용하면 4등급으로 나눠지고 숫자 5를 사용하면 다음과 같이 5등급 으로 나눠집니다.



데이터 분석 함수로 등급 출력하기

- ORDER BY SAL DESC에서 NULLS LAST는 월급을 높은 것부터 출력되도록 정렬하는데, NULL을 맨 아래에 출력하겠다는 의미입니다.
- 다음의 두 쿼리 결과를 비교해봅니다.

NULLS LAST를 사용 안 했을 때

```
SELECT ename, comm
  FROM emp
  WHERE deptno = 30
  ORDER BY comm DESC;
```

ENAME	COMM
BLAKE	
JAMES	
MARTIN	1400
WARD	500
ALLEN	300
TURNER	0

NULLS LAST를 사용했을 때

```
SELECT ename, comm
  FROM emp
  WHERE deptno = 30
  ORDER BY comm DESC NULLS LAST;
```

ENAME	COMM
BLAKE	1400
JAMES	500
MARTIN	300
WARD	0
ALLEN	
TURNER	

- NULLS LAST를 사용했을 때는 NULL 값이 마지막으로 정렬됩니다.

데이터 분석 함수로 순위의 비율 출력하기

- 이름과 월급, 월급의 순위, 월급의 순위 비율을 출력해 보겠습니다.

```
1  SELECT ename, sal, RANK() over (order by sal desc) as RANK ,
2                     DENSE_RANK() over (order by sal desc) as DENSE_RANK,
3                     CUME_DIST()  over (order by sal desc) as CUM_DIST
4    FROM emp;
```

- 위의 결과에서 순위는 1등부터 14등까지 있습니다.
- 사원 KING의 CUME_DIST 열의 첫번째 행 0.0714285기인 것은 1/14로 계산된 결과입니다.
- 전체 14등 중 1등의 비율인 0.07로 출력되었습니다.
- 2등은 2명이어서 3/14로 계산되어 0.214285714로 출력되었습니다.
- 같은 등수가 없고 그 등수에 해당하는 사원이 한 명이면 전체 등수로 해당 등수를 나눠서 계산하고, 같은 등수가 여러 명 있으면 여러 명 중 마지막 등수로 계산합니다.
- 위의 결과처럼 2등이 2명이므로 3을 14로 나눈 것입니다.
- 2등이 3명이었다면 4를 14로 나눠서 출력합니다.

데이터 분석 함수로 순위의 비율 출력하기

- 다음은 PARTITION BY JOB을 사용해 직업별로 CUME_DIST를 출력하는 예제입니다.

```
1  SELECT job, ename, sal, RANK() over ( partition by job
2                                        order by sal desc) as RANK ,
3                          CUME_DIST() over ( partition by job
4                                        order by sal desc) as CUM_DIST
5    FROM emp;
```

데이터 분석 함수로 데이터를 가로로 출력하기

- 부서 번호를 출력하고, 부서 번호 옆에 해당 부서에 속하는 사원들의 이름을 가로로 출력해 보겠습니다.

```
1  SELECT deptno, LISTAGG(ename,',') within group (order by ename) as EMPLOYEE
2    FROM emp;
3    GROUP BY deptno;
```

- LISTAGG 함수는 데이터를 가로로 출력하는 함수입니다.
- LISTAGG에 구분자로 콤마(,)를 사용하여 이름이 콤마(,)로 구분되었습니다.
- 만약 콤마(,)가 아니라 슬래쉬(/)로 구분하고자 하면 슬래쉬(/)를 사용하여 구분할 수 있습니다.
- WITHIN GROUP은 '~이내의,라는 뜻으로 group 다음에 나오는 괄호에 속한 그룹의 데이터를 출력하겠다는 것입니다.
- Order by ename으로 이름이 A, B, C, D 순으로 정렬되어 출력되었습니다.
- GROUP BY절은 LISTAGG 함수를 사용하려면 필수로 기술해야 하는 절입니다.

데이터 분석 함수로 데이터를 가로로 출력하기

- 다음의 예제는 직업과 그 직업에 속한 사원들의 이름을 가로로 출력하는 예제입니다.

```
1  SELECT job, LISTAGG(ename,',') within group (ORDER BY ename asc) as employee
2    FROM emp;
3    GROUP BY job;
```

- 이름 옆에 월급도 같이 출력하려면 연결 연산자를 사용하여 작성합니다.
- 각 직업별로 월급의 분포가 어떻게 되는지 한눈에 확인할 수 있게 되었습니다.

```
1  SELECT job,
2        LISTAGG(ename||'('||sal||')',',') within group (ORDER BY ename asc) as employee
3    FROM emp
4    GROUP BY job;
```

데이터 분석 함수로 바로 전 행과 다음 행 출력하기

- 사원 번호, 이름, 월급을 출력하고 그 옆에 바로 전 행의 월급을 출력하고, 또 옆에 바로 다음 행의 월급을 출력해 보겠습니다.

```
1  SELECT empno, ename, sal,
2         LAG(sal,1) over (order by sal asc) "전 행",
3         LEAD(sal,1) over (order by sal asc) "다음 행"
4    FROM emp
5    WHERE job in ('ANALYST','MANAGER');
```

- LAG 함수는 바로 전 행의 데이터를 출력하는 함수입니다.
- 숫자 1을 사용하면 바로 전 행이 출력됩니다.
- 숫자 2를 사용하면 바로 전 전 행이 출력됩니다.
- LEAD 함수는 바로 다음 행의 데이터를 출력하는 함수입니다.
- 숫자 1을 사용하면 바로 다음 행이 출력됩니다.
- 숫자 2를 사용하면 바로 다음 다음 행이 출력됩니다.

데이터 분석 함수로 바로 전 행과 다음 행 출력하기

- 직업이 ANALYST 또는 MANAGER인 사원들의 사원 번호, 이름, 입사일, 바로 전에 입사한 사원의 입사일, 바로 다음에 입사한 사원의 입사일을 출력해 보겠습니다.

```
1  SELECT empno, ename, hiredate,
2       LAG(hiredate,1) over (order by hiredate asc) "전 행",
3       LEAD(hiredate,1) over (order by hiredate asc) "다음 행"
4    FROM emp
5    WHERE job in ('ANALYST','MANAGER');
```

- 다음의 예제는 부서 번호, 사원 번호, 이름, 입사일, 바로 전에 입사한 사원의 입사일을 출력하고 바로 다음에 입사한 사원의 입사일을 출력하는데, 부서 번호별로 구분해서 출력하는 쿼리입니다.

```
1  SELECT deptno, empno, ename, hiredate,
2       LAG(hiredate,1) over ( partition by deptno
3                              order by hiredate asc) "전 행",
4       LEAD(hiredate,1) over ( partition by deptno
5                              order by hiredate asc) "다음 행"
6    FROM emp;
```

COLUMN을 ROW로 출력하기 ①

- 부서 번호, 부서 번호별 토탈 월급을 출력하는데, 다음과 같이 가로로 출력해 보겠습니다.

```
1  SELECT SUM(DECODE(deptno, 10, sal)) as "10",
2         SUM(DECODE(deptno, 20, sal)) as "20",
3         SUM(DECODE(deptno, 30, sal)) as "30"
4    FROM emp;
```

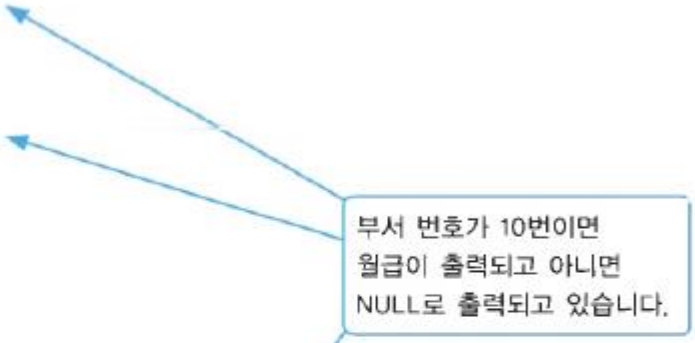
- 어떻게 가로로 출력했는지 하나씩 살펴보겠습니다.
- 먼저 다음의 쿼리는 부서 번호가 10번이면 월급이 출력되고 아니면 NULL이 출력 되는 예제입니다.

```
1  SELECT deptno, DECODE(deptno,10, sal) as "10"
2    FROM emp;
```

COLUMN을 ROW로 출력하기 ①

- 출력 결과

DEPTNO	10
10	5000
30	
10	2450
20	
30	
30	
30	
30	
30	
20	
20	
20	
20	
10	1300



COLUMN을 ROW로 출력하기 ①

- 위의 결과에서 DEPTNO 컬럼을 제외하고 DECODE(depmo,10, sal)만 출력한 다음 출력된 결과 값을 다 더해서 출력해 보겠습니다.

```
1   SELECT SUM(DECODE(deptno,10, sal)) as "10"
2     FROM emp;
```

- 여기에 20번과 30번도 같이 출력하면 다음의 쿼리가 됩니다.

```
1   SELECT SUM(DECODE(deptno, 10, sal)) as "10",
2          SUM(DECODE(deptno, 20, sal)) as "20",
3          SUM(DECODE(deptno, 30, sal)) as "30"
4     FROM emp;
```

COLUMN을 ROW로 출력하기 ①

- 다음 쿼리는 직업, 직업별 토탈 월급을 출력하는데 아래처럼 가로로 출력하는 예제입니다.

```
1   SELECT SUM(DECODE(job,'ANALYST',sal)) as "ANALYST",
2          SUM(DECODE(job,'CLERK',sal))  as "CLERK",
3          SUM(DECODE(job,'MANAGER',sal)) as "MANAGER",
4          SUM(DECODE(job,'SALESMAN',sal)) as "SALESMAN"
5     FROM emp;
```

- 직업이 ANALYST이면 월급을 출력하고 그 월급을 SUM 함수로 합산하면 하나의 값이 출력됩니다.
- 나머지 직업도 같은 방법으로 합산하여 결과를 출력합니다.
- 위의 쿼리는 사원 테이블에 직업이 ANALYST, CLERK, MANAGER, SALESMAN이 있다는 것을 안다는 가정 하에 작성된 쿼리입니다.
- 모른다고 가정하고 출력되게 하려면 PL/SQL을 사용해야 합니다.

COLUMN을 ROW로 출력하기 ①

- 다음의 예제는 위의 예제의 SELECT절에 DEPTNO를 추가하였습니다.
- DEPTNO를 추가한 이유는 부서 번호별로 각각 직업의 토탈 월급의 분포를 보기 위함입니다.
- DEPTNO를 그룹 함수와 같이 나열하였으므로 GROUP BY deptno를 추가하여 실행합니다.

```
1  SELECT DEPTNO, SUM(DECODE(job,'ANALYST',sal)) as "ANALYST",
2         SUM(DECODE(job,'CLERK',sal))  as "CLERK",
3         SUM(DECODE(job,'MANAGER',sal)) as "MANAGER",
4         SUM(DECODE(job,'SALESMAN',sal)) as "SALESMAN"
5    FROM emp
6    GROUP BY deptno;
```

- 출력 결과를 보면 CLERK과 MANAGER는 모든 부서에 존재하지만, ANALYST는 20번에만 존재하고, SALESMAN은 30번에만 존재한다는 것을 알 수 있습니다.

COLUMN을 ROW로 출력하기 ②

- 부서 번호, 부서 번호별 토탈 월급을 Pivot문을 사용하여 가로로 출력해 보겠습니다.

```
1  SELECT *
2    FROM (select deptno, sal from emp)
3    PIVOT (sum(sal) for deptno in (10,20,30));
```

- 앞 절에서 SUM과 DECODE를 이용해 출력한 결과를 PIVOT문을 이용하면 좀더 간단한 쿼리문으로 출력할 수 있습니다.
- 부서 번호와 부서 번호별 토탈 월급을 출력하기 위해 필요한 데이터는 부서 번호와 월급 뿐입니다.
- 그래서 FROM절에 EMP 테이블만 쓰지 않고 쿼리문을 사용하여 EMP 테이블에서 부서 번호와 월급만 조회해 왔습니다.
- 위의 출력 결과를 위해 직업이나 입사일은 필요하지 않습니다.
- 오직 부서 번호와 월급만 필요합니다.
- 그래서 FROM절에 괄호를 쓰고 괄호 안에 부서 번호와 월급만 조회하는 쿼리문을 넣었습니다.
- 그리고 출력되는 부서 번호와 월급은 이 전체 쿼리문에서 마치 테이블처럼 사용됩니다.

COLUMN을 ROW로 출력하기 ②

- FROM절에서 추출한 부서 번호와 월급을 가지고 부서 번호별 토탈 월급을 출력합니다.
- 부서번호는 10번, 20번, 30번에 대한 것을 출력합니다.

```
SELECT *
  FROM (select deptno, sal from emp)
  PIVOT (sum(sal) for deptno in (10,20,30));
```

DEPTNO	SAL
10	5000
30	2850
10	2450
20	2975
30	1250
30	1600
30	1500
30	950
30	1250
20	3000
20	800
20	3000
20	1100
10	1300

10	8750
20	10875
30	9400

회전

10	20	30
8750	10875	9400

COLUMN을 ROW로 출력하기 ②

- 이번에는 문자형 데이터를 다뤄보겠습니다.
- 다음의 쿼리는 PIVOT문을 이용해서 직업과 직업별 토탈 월급을 가로로 출력하는 예제입니다.

```
1  SELECT *
2    FROM (select job, sal from emp)
3    PIVOT (sum(sal) for job in ('ANALYST', 'CLERK','MANAGER', 'SALESMAN'));
```

- 출력되는 결과에 필요한 데이터가 있는 컬럼인 직업과 월급을 선택합니다.
- 위의 결과에서 필요한 컬럼은 직업과 월급뿐입니다.
- PIVOT문을 사용할 때는 FROM절에 괄호를 사용해서 특정 컬럼만 선택해야 합니다.
- FROM emp라고 작성하면 에러가 발생합니다.
- 반드시 괄호 안에 결과에 필요한 컬럼만 선택하는 쿼리문을 작성해야 합니다.

COLUMN을 ROW로 출력하기 ②

- PIVOT문을 이용해서 토탈 월급을 출력합니다.
- in 다음에 토탈 월급을 출력할 직업 리스트를 나열합니다.
- 그런데 컬럼명에 싱글 쿼테이션 마크가 붙어서 출력되었습니다.
- 싱글 쿼테이션 마크가 출력되지 않게 하려면 다음과 같이 as 뒤에 해당 직업을 더블 쿼테이션으로 둘러서 작성하게 되면 싱글 쿼테이션 마크 없이 출력될 수 있게 됩니다.

```
1  SELECT *
2    FROM (select job, sal from emp)
     PIVOT (sum(sal) for job in ('ANALYST' as "ANALYST", 'CLERK' as "CLERK",
3                                'MANAGER' as "MANAGER", 'SALESMAN' as "SALESMAN"));
```

ROW를 COLUMN으로 출력하기

- 실습에 앞서 먼저 order2 테이블을 생성하겠습니다.

```
1   drop  table order2;
2   create table order2
3   ( ename  varchar2(10),
4     bicycle  number(10),
5     camera   number(10),
6     notebook  number(10) );
7
8   insert  into  order2  values('SMITH', 2,3,1);
9   insert  into  order2  values('ALLEN',1,2,3 );
10  insert  into  order2  values('KING',3,2,2 );
11
12  commit;
```

ROW를 COLUMN으로 출력하기

- UNPIVOT문을 사용하여 컬럼을 로우로 출력해 보겠습니다.

```
1  SELECT *
2    FROM order2
3    UNPIVOT (건수 for 아이템 in (BICYCLE, CAMERA, NOTEBOOK));
```

- UNPIVOT문은 PIVOT문과는 반대로 열을 행으로 출력합니다.
- 테이블 order2의 열인 BICYCLE, CAMERA, NOTEBOOK이 출력 결과에서 행으로 출력되고 있습니다.

ORDER2 테이블

ENAME	BICYCLE	CAMERA	NOTEBOOK
SMITH	2	3	1
ALLEN	1	2	3
KING	3	2	2

결과

ENAME	아이템	건수
SMITH	BICYCLE	2
SMITH	CAMERA	3
SMITH	NOTEBOOK	1
ALLEN	BICYCLE	1
ALLEN	CAMERA	2
ALLEN	NOTEBOOK	3
KING	BICYCLE	3
KING	CAMERA	2
KING	NOTEBOOK	2

ROW를 COLUMN으로 출력하기

- '건수'는 가로로 저장되어 있는 데이터를 세로로 unpivot시킬 출력 열 이름입니다.
- 이 열 이름은 임의로 지정하면 됩니다.
- for 다음에 '아이템'은 가로로 되어 있는 order2 테이블의 컬럼명을 unpivot시켜 세로로 출력할 열 이름입니다.
- 이 열 이름도 임의로 지정하면 됩니다.
- 다음과 같이 AS절을 사용하여 값을 변경할 수 있습니다.

```
1  SELECT *
2    FROM order2
3    UNPIVOT (건수 for 아이템 in (BICYCLE as 'B', CAMERA as 'C', NOTEBOOK as 'N'));
```

ORDER2 테이블

ENAME	BICYCLE	CAMERA	NOTEBOOK
SMITH	2	3	1
ALLEN	1	2	3
KING	3	2	2

결과

ENAME	아이템	건수
SMITH	B	2
SMITH	C	3
ALLEN	B	1
ALLEN	C	2
ALLEN	N	3
KING	B	3
KING	C	2

ROW를 COLUMN으로 출력하기

- order2 테이블의 데이터에 NULL이 포함되어 있었다면 UNPIVOT된 결과에서 출력이 되지 않습니다.
- 다음의 UPDATE문을 실행하여 SMITH의 NOTEBOOK을 NULL로 변경하고 UNPIVOT문을 다시 실행해 봅니다.

```
1  UPDATE ORDER2 SET NOTEBOOK=NULL WHERE ENAME='SMITH';
2
3  SELECT *
4     FROM order2;
```

ORDER2 테이블

NULL로 변경되었습니다.

ENAME	BICYCLE	CAMERA	NOTEBOOK
SMITH	2	3	
ALLEN	1	2	3
KING	3	2	2

결과

ENAME	아이템	건수
SMITH	B	2
SMITH	C	3
ALLEN	B	1
ALLEN	C	2
ALLEN	N	3
KING	B	3
KING	C	2
KING	N	2

ROW를 COLUMN으로 출력하기

- SMITH의 NOTEBOOK 정보가 출력되지 않았습니다.
- 이럴 때 NULL 값인 행도 결과에 포함시키려면 다음과 같이 INCLUDE NULLS를 사용합니다.

```
1  SELECT *
2     FROM order2
3     UNPIVOT INCLUDE NULLS (건수 for 아이템 in (BICYCLE as 'B', CAMERA as 'C', NOTEBOOK as 'N'));
```

데이터 분석 함수로 누적 데이터 출력하기

- SMITH의 NOTEBOOK 정보가 출력되지 않았습니다.
- 이럴 때 NULL 값인 행도 결과에 포함시키려면 다음과 같이 INCLUDE NULLS를 사용합니다.

```
1  SELECT empno, ename, sal, SUM(SAL) OVER (ORDER BY empno ROWS
2                                          BETWEEN UNBOUNDED PRECEDING
3                                          AND CURRENT ROW) 누적치
4    FROM emp
5    WHERE job in ('ANALYST','MANAGER');
```

- 위의 예제는 직업이 ANALYST, MANAGER인 사원들의 사원 번호, 이름, 월급, 월급의 누적치를 출력하는 쿼리입니다.
- OVER 다음의 괄호 안에는 값을 누적할 윈도우를 지정할 수 있습니다.
- ORDER BY empno를 통해 사원 번호를 오름차순으로 정렬을 하고 정렬된 것을 기준으로 월급의 누적치를 출력합니다.

데이터 분석 함수로 누적 데이터 출력하기

- UNBOUNDED PRECEDING은 제일 첫 번째 행을 가리킵니다.
- 제일 첫 번째 행의 값은 2975입니다.
- BETWEEN UNBOUNDED AND CURRENT ROW는 제일 첫 번째 행부터 현재 행까지의 값을 말합니다.
- 두 번째 행의 누적치 5825는 제일 첫 번째 행의 값인 2975와 현재 행의 값인 2850을 합계한 결과입니다.

윈도우 기준	윈도우 방식	설명
ROWS	UNBOUNDED PRECEDING	맨 첫 번째 행을 가리킵니다.
	UNBOUNDED FOLLOWING	맨 마지막 행을 가리킵니다.
	CURRENT ROW	현재 행을 가리킵니다.

EMPNO	ENAME	SAL	누적치
7566	JONES	2975	2450
7698	BLAKE	2850	5825
7782	CLARK	2450	8275
7788	SCOTT	3000	11275
7902	FORD	3000	14275

데이터 분석 함수로 비율 출력하기

- 부서 번호가 20번인 사원들의 사원 번호, 이름, 월급을 출력하고, 20번 부서 번호 내에서 자신의 월급 비율이 어떻게 되는지 출력해 보겠습니다.

```
1   SELECT empno, ename, sal, RATIO_TO_REPORT(sal) OVER () as 비율
2     FROM emp
3     WHERE deptno = 20;
```

- 20번인 사원들의 월급의 합계는 10875입니다.
- 첫 번째 행인 JONES의 월급이 20번 전체 월급에서 차지하는 비율은 2975/10875인 0.273563218입니다.
- 다음의 쿼리로 확인해 보겠습니다.

```
1   SELECT empno, ename, sal, RATIO_TO_REPORT(sal) OVER () as 비율,
2                             SAL/SUM(sal) OVER () as "비교 비율"
3     FROM emp
4     WHERE deptno = 20;
```

- 20번 부서 번호인 사원들의 월급을 20번 부서 번호인 사원들의 전체 월급으로 나누어 출력합니다.
- RAITIO_TO_REPORT(sal)의 결과와 동일하게 출력됩니다.

데이터 분석 함수로 집계 결과 출력하기 ①

- 직업과 직업별 토탈 월급을 출력하는데, 맨 마지막 행에 토탈 월급을 출력해 보겠습니다.

```
1  SELECT job, sum(sal)
2    FROM emp
3    GROUP BY ROLLUP(job);
```

- 위의 예제는 ROLLUP을 이용하여 직업과 직업별 토탈 월급을 출력하고 맨 아래쪽에 전체 토탈 월급을 추가적으로 출력하는 쿼리입니다.
- 직업과 직업별 토탈 월급을 출력하는 쿼리에 ROLLUP 만 붙여주면 전체 토탈 월급을 추가적으로 볼수 있습니다.
- ROLLUP을 사용하면 맨 아래에 토탈 월급도 출력되고 JOB 컬럼의 데이터도 오름차순으로 정렬되어 출력됩니다.
- 다음의 쿼리는 ROLLUP에 컬럼을 두 개를 사용한 경우입니다.

```
1  SELECT deptno, job, sum(sal)
2    FROM emp
3    GROUP BY ROLLUP(deptno, job);
```

데이터 분석 함수로 집계 결과 출력하기 ①

- 세 가지 집계 결과가 출력되었습니다.
- ROLLUP(deptno, job)으로 ROLLUP에 컬럼을 2개 사용하면 집계 결과는 컬럼의 개수(2개) + 1로 3개가 출력됩니다.

GROUP BY ROLLUP(deptno,job)

집계 결과	1	deptno, job	부서 번호별 직업별 토탈 월급
	2	deptno	부서 번호별 토탈 월급
	3	()	전체 토탈 월급

- 집계된 결과가 왜 세 가지가 되는지 쉽게 이해하는 방법은 다음과 같습니다.
- 먼저 ROLLUP 안에 있는 컬럼들을 그대로 적고 뒤에서부터 하나씩 제거해 나갑니다.

집계 결과	1	deptno, job	deptno, job
	2	deptno	deptno
	3	()	

데이터 분석 함수로 집계 결과 출력하기 ①

- 위와 같이 ROLLUP 함수 안에 컬럼명을 전부 기술하고 뒤에 있는 컬럼부터 하나씩 지워 나가면 예상되는 집계 결과의 개수를 예측할 수 있습니다.
- 다음의 컬럼 3개 집계 결과를 예상해 보겠습니다.

GROUP BY ROLLUP(deptno, job, mgr)

집계 결과	1	deptno, job, mgr	deptno, job, mgr
	2	deptno, job	deptno, job
	3	deptno	deptno
	4	()	

- ROLLUP 함수 안에 컬럼이 3개이므로 4개의 집계 결과가 출력됩니다.

데이터 분석 함수로 집계 결과 출력하기 ②

- 직업, 직업별 토탈 월급을 출력하는데, 첫 번째 행에 토탈 월급을 출력해 보겠습니다.

```
1  SELECT job, sum(sal)
2    FROM emp
3    GROUP BY CUBE(job);
```

- CUBE를 사용했을 때와 사용하지 않았을 때의 차이는 다음과 같습니다.

CUBE를 추가하지 않았을 때

DEPTNO	SUM(SAL)
30	
10	
20	

CUBE를 추가했을 때

DEPTNO	SUM(SAL)
	29025
10	8750
20	10875
30	9400

- CUBE를 추가했을 때 맨 위쪽에 전체 토탈 월급이 추가가 되었고 부서 번호도 오름차순으로 정렬이 되어 출력되었습니다.

데이터 분석 함수로 집계 결과 출력하기 ②

- 다음은 CUBE에 컬럼 2개를 사용한 쿼리입니다.

```
1  SELECT deptno, job, sum(sal)
2    FROM emp
3    GROUP BY CUBE(deptno,job);
```

DEPTNO	JOB	SUM(SAL)	
		29025	← 토탈 월급이 출력되고 있습니다.
	CLERK	4150	
	ANALYST	6000	
	MANAGER	8275	직업별 토탈 월급이 출력되고 있습니다.
	SALESMAN	5600	
	PRESIDENT	5000	
10		8750	← 부서 번호별 토탈 월급이 출력되고 있습니다.
10	CLERK	1300	
10	MANAGER	2450	직업별 부서 번호별 토탈 월급이 출력되고 있습니다.
10	PRESIDENT	5000	
20		10875	← 부서 번호별 토탈 월급이 출력되고 있습니다.

데이터 분석 함수로 집계 결과 출력하기 ②

- GROUP BY CUBE(deptno, job)는 총 4가지 경우 집계 결과가 나옵니다.
- 전체 토탈 월급이 맨 위에 출력되고 그 다음으로 직업별 토탈 월급이 출력됩니다.
- 부서 번호별 토탈 월급과 부서 번호별 직업별 토탈 월급이 그 다음으로 출력되면서 총 4가지 집계 결과가 출력되었습니다.
- 정리하면 다음과 같습니다.

집계 결과	1	deptno, job	부서 번호별 직업별 토탈 월급
	2	deptno	부서 번호별 토탈 월급
	3	job	직업별 토탈 월급
	4	()	전체 토탈 월급

- GROUP BY ROLLUP(deptno, job)과 비교하면 job에 대한 집계 한 가지가 더 출력되었습니다.

데이터 분석 함수로 집계 결과 출력하기 ③

- 부서 번호와 직업, 부서 번호별 토탈 월급과 직업별 토탈 월급, 전체 토탈 월급을 출력해 보겠습니다.

```
1   SELECT deptno, job, sum(sal)
2     FROM emp
3     GROUP BY GROUPING SETS((deptno), (job), () );
```

- GROUPING SETS는 앞에서 배웠던 ROLLUP과 CUBE보다 집계된 결과를 예상하기 더 쉽습니다.
- 왜냐하면 GROUPING SETS에 집계하고 싶은 컬럼들을 기술하면 그대로 출력되기 때문입니다.
- GROUPING SETS 괄호 안에 다음과 같이 집계하고 싶은 컬럼명을 기술하면, 기술한대로 결과를 출력해주면 됩니다.

GROUPING SETS	출력 결과
GROUPING SETS((deptno), (job), ())	부서 번호별 집계, 직업별 집계, 전체 집계
GROUPING SETS((deptno), (job))	부서 번호별 집계, 직업별 집계
GROUPING SETS((deptno,job),())	부서 번호와 직업별 집계, 전체 집계
GROUPING SETS((deptno,job))	부서 번호와 직업별 집계

데이터 분석 함수로 집계 결과 출력하기 ③

- 위의 문법에 따라 다음의 ROLLUP문을 GROUPING SETS로 변경하면 다음과 같습니다.

ROLLUP을 사용했을 때

```
SELECT deptno, sum(sal)
 FROM emp
 GROUP BY ROLLUP(deptno);
```

DEPTNO	SUM(SAL)
10	8750
20	10875
30	9400
	29025

GROUPING SETS를 사용했을 때

```
SELECT deptno, sum(sal)
  FROM emp
  GROUP BY GROUPING
SETS((deptno), () );
```

DEPTNO	SUM(SAL)
10	9400
20	10875
30	9400
	29025

- 괄호 ()는 전체를 말합니다.
- 전체를 대상으로 월급을 집계합니다.
- 두 가지 모두 결과는 동일하지만, GROUPING SETS가 결과를 예측하기가 더 쉽습니다.

데이터 분석 함수로 출력 결과 넘버링 하기

- ROW_NUMBER()는 출력되는 각 행에 고유한 숫자 값을 부여하는 데이터 분석 함수입니다.
- PSEUDOCOLUMN 인 ROWNUM과 유사하며 RANK와 DENSE_RANK 분석 함수와는 다릅니다.

```
1  SELECT empno, ename, sal, RANK() OVER (ORDER BY sal DESC) RANK,
2                            DENSE_RANK() OVER (ORDER BY sal DESC) DENSE_RANK,
3                            ROW_NUMBER() OVER (ORDER BY sal DESC) 번호
4    FROM emp
5    WHERE deptno = 20;
```

- 출력 결과를 보면 첫 번째 행인 FORD와 두 번째 행인 SCOTT의 월급이 서로 같아 RANK와 DENSE_RANK는 순위를 둘 다 1로 출력하고 있으나, ROW_NUMBER는 1, 2로 출력하고 있습니다.
- ROW_NUMBER()는 출력되는 결과에 번호를 순서대로 부여해서 출력합니다.

데이터 분석 함수로 출력 결과 넘버링 하기

- ROW_NUMBER() 함수는 OVER 다음 괄호 안에 반드시 ORDER BY절을 기술해야 합니다.
- 그렇지 않으면 다음과 같이 에러가 발생합니다.

```
1  SELECT empno, ename, sal, ROW_NUMBER() OVER () 번호
2    FROM emp
3    WHERE deptno = 20;
```

출력 결과

```
SELECT empno, ename, sal, ROW_NUMBER() OVER () 번호
                          *
1행 오류:
ORA-30485: 윈도우 지정에 ORDER BY 표현식이 없습니다.
```

- 다음의 쿼리는 부서 번호별로 월급에 대한 순위를 출력하는 쿼리입니다.
- PARTITION BY를 사용하여 부서 번호별로 파티션해서 순위를 부여합니다.

```
1  SELECT deptno, ename, sal, ROW_NUMBER() OVER( PARTITION BY deptno
2                                                ORDER BY sal DESC) 번호
3    FROM emp
5    WHERE deptno in (10,20);
```