Structured Query Language (SQL)

Part 4. Group and Window Functions

Hyeokman Kim
School of Computer Science
Kookmin Univ.

ANSI/ISO SQL

SQL DML : GROUP FUNCTIONS

데이터 분석을 위한 **ANSI/SQL Functions**

- Aggregate Functions
  - COUNT, SUM, AVG, MIN, MAX, STDDEV

- Group Functions
  - 소계, 중계, 합계, 총계 등을 위해, 하나의 SQL로 “테이블을 한 번만 읽어서” 빠르게 원하는 리포트를 작성하도록 도와주는 함수.
    - 기존에는 레벨별 집계를 위한 여러 단계의 SQL을 UNION, UNION ALL로 묶은 후 하나의 테이블을 여러 번 읽어 다시 재정렬하는 복잡한 단계를 거쳐야 가능한 작업임.
  - 종류
    - ROLLUP : 소그룹 간의 소계를 계산.
    - CUBE : GROUP BY 항목들 간 다차원적인 소계를 계산.
    - GROUPING SETS : 특정 항목에 대한 소계를 계산.

- Window Functions (Analytic 혹은 Rank functions)
  - 행과 행간의 관계를 쉽게 정의하기 위해 만든 함수.

1. ROLLUP 함수

- 레벨별 소계(subtotal)를 계산하기 위한 함수
  - 함수의 인수로 grouping column list를 가짐.
  - grouping column list 에 소계에 필요한 컬럼을 나열함.
  - Grouping columns의 수가 N이면, N+1 level의 subtotal이 생성됨.

- 주의
  - ROLLUP의 인수는 계층 구조이므로, 인수 순서가 바뀌면 수행 결과도 바뀌게 되므로 인수의 순서에도 주의.
  - 결과에 대한 정렬이 필요한 경우는 ORDER BY 절에 정렬 칼럼을 명시해야 함.

예제

□ Step 1: 부서명과 업무명을 기준으로, 사원수와 급여 합을 집계.

```
SELECT   DNAME, JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY DNAME, JOB
```

DNAME	JOB	Total Empl	Total Sal
SALES	MANAGER	1	2850
SALES	CLERK	1	950
ACCOUNTING	MANAGER	1	2450
RESEARCH	ANALYST	2	6000
ACCOUNTING	CLERK	1	1300
SALES	SALESMAN	4	5600
RESEARCH	MANAGER	1	2975
ACCOUNTING	PRESIDENT	1	5000
RESEARCH	CLERK	2	1900

9개의 행이 선택되었다.

ㅁ Step 1': 부서명과 업무명을 기준으로, 사원수와 급여 합을 집계.

```
SELECT   DNAME, JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY DNAME, JOB
ORDER    BY DNAME, JOB;
```

DNAME	JOB	Total Empl	Total Sal
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT	1	5000
RESEARCH	ANALYST	2	6000
RESEARCH	CLERK	2	1900
RESEARCH	MANAGER	1	2975
SALES	CLERK	1	950
SALES	MANAGER	1	2850
SALES	SALESMAN	4	5600

9개의 행이 선택되었다.

□ Step 2: ROLLUP 함수 사용

```
SELECT   DNAME, JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY ROLLUP(DNAME, JOB);
```

– 3개 레벨에 대에 집계 함수 실행

- ◆ L1 레벨: GROUP BY 이외의 컬럼에 대해, subtotal이 계산되지 않은 레벨 (9건)
- ◆ L2 레벨: DNAME 별 subtotal (3건)
- ◆ L3 레벨: grand total (1건)

DNAME	JOB	Total Empl	Total Sal
SALES	CLERK	1	950
SALES	MANAGER	1	2850
SALES	SALESMAN	4	5600
SALES		6	9400
RESEARCH	CLERK	2	1900
RESEARCH	ANALYST	2	6000
RESEARCH	MANAGER	1	2975
RESEARCH		5	10875
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT	1	5000
ACCOUNTING		3	8750
		14	29025

13개의 행이 선택되었다.

계층간 정렬은 제공하나,
계층내 정렬을 제공안함.

◻ Step 2': ROLLUP 함수 사용, 계층내 정렬

```
SELECT   DNAME, JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY ROLLUP(DNAME, JOB)
ORDER    BY DNAME, JOB;
```

DNAME	JOB	Total Empl	Total Sal
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT	1	5000
ACCOUNTING		3	8750
RESEARCH	ANALYST	2	6000
RESEARCH	CLERK	2	1900
RESEARCH	MANAGER	1	2975
RESEARCH		5	10875
SALES	CLERK	1	950
SALES	MANAGER	1	2850
SALES	SALESMAN	4	5600
SALES		6	9400
		14	29025

13개의 행이 선택되었다.

◻ Step 3: GROUPING 함수 사용 (subtotal이 계산되면 1, 아니면 0)

```
SELECT   DNAME, GROUPING(DNAME),
         JOB, GROUPING(JOB),
         COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY ROLLUP(DNAME, JOB)
ORDER    BY DNAME, JOB;
```

DNAME	GROUPING(DNAME)	JOB	GROUPING(JOB)	Total Empl	Total Sal
SALES	0	CLERK	0	1	950
SALES	0	MANAGER	0	1	2850
SALES	0	SALESMAN	0	4	5600
SALES	0		1	6	9400
RESEARCH	0	CLERK	0	2	1900
RESEARCH	0	ANALYST	0	2	6000
RESEARCH	0	MANAGER	0	1	2975
RESEARCH	0		1	5	10875
ACCOUNTING	0	CLERK	0	1	1300
ACCOUNTING	0	MANAGER	0	1	2450
ACCOUNTING	0	PRESIDENT	0	1	5000
ACCOUNTING	0		1	3	8750
	1		1	14	29025

13개의 행이 선택되었다.

□ Step 4: GROUPING 함수 + CASE 절 사용

```
SELECT   CASE GROUPING(DNAME) WHEN 1 THEN 'All Departments'
                                      ELSE DNAME END AS DNAME,
         CASE GROUPING(JOB) WHEN 1 THEN 'All Jobs' ELSE JOB END AS JOB,
         COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY ROLLUP(DNAME, JOB);
```

DNAME	JOB	Total Empl	Total Sal
SALES	CLERK	1	950
SALES	MANAGER	1	2850
SALES	SALESMAN	4	5600
SALES	All Jobs	6	9400
RESEARCH	CLERK	2	1900
RESEARCH	ANALYST	2	6000
RESEARCH	MANAGER	1	2975
RESEARCH	All Jobs	5	10875
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT	1	5000
ACCOUNTING	All Jobs	3	8750
All Departments	All Jobs	14	29025

13개의 행이 선택되었다.

□ ☞ 한 단계 subtotal만 필요할 때

```
SELECT   CASE GROUPING(DNAME) WHEN 1 THEN 'All Departments'
                                      ELSE DNAME END AS DNAME,
         CASE GROUPING(JOB) WHEN 1 THEN 'All Jobs' ELSE JOB END AS JOB,
         COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY DNAME, ROLLUP(JOB);
```

DNAME	JOB	Total Empl	Total Sal
SALES	CLERK	1	950
SALES	MANAGER	1	2850
SALES	SALESMAN	4	5600
SALES	All Jobs	6	9400
RESEARCH	CLERK	2	1900
RESEARCH	ANALYST	2	6000
RESEARCH	MANAGER	1	2975
RESEARCH	All Jobs	5	10875
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT	1	5000
ACCOUNTING	All Jobs	3	8750

12개의 행이 선택되었다.

□ ☞ 부서별, (JOB & MGR)별 소계가 필요할 때

```
SELECT   DNAME, JOB, MGR, SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY ROLLUP(DNAME, (JOB, MGR));
```

DNAME	JOB	MGR	Total Sal
SALES	CLERK	7698	950
SALES	MANAGER	7839	2850
SALES	SALESMAN	7698	5600
SALES			9400
RESEARCH	CLERK	7788	1100
RESEARCH	CLERK	7902	800
RESEARCH	ANALYST	7566	6000
RESEARCH	MANAGER	7839	2975
RESEARCH			10875
ACCOUNTING	CLERK	7782	1300
ACCOUNTING	MANAGER	7839	2450
ACCOUNTING	PRESIDENT		5000
ACCOUNTING			8750
			29025

14개의 행이 선택되었다.

2. CUBE 함수

- grouping column list 의 모든 경우에 대해 소계(subtotal)를 계산.
  - 함수의 인수로 grouping column list를 가짐.
  - grouping column list 에 소계에 필요한 컬럼을 나열함.
  - Grouping column의 수가 N이면, 2^N level의 subtotal이 생성됨.

- 주의
  - CUBE의 인수는 계층 구조가 아님. 즉, 인수의 순서가 바뀌는 경우, 투플 간에 정렬 순서는 바뀔 수 있어도 데이터 결과는 같음.
  - 결과에 대한 정렬이 필요한 경우는 ORDER BY 절에 정렬 컬럼을 명시해야 함.

□ 예제: 앞의 step 4에서 ROLLUP 대신 CUBE를 사용

```
SELECT   CASE GROUPING(DNAME) WHEN 1 THEN 'All Departments'
                                      ELSE DNAME END AS DNAME,
         CASE GROUPING(JOB) WHEN 1 THEN 'All Jobs' ELSE JOB END AS JOB,
         COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY CUBE(DNAME, JOB);
```

DNAME	JOB	Total Empl	Total Sal
All Departments	All Jobs	14	29025
All Departments	CLERK	4	4150
All Departments	ANALYST	2	6000
All Departments	MANAGER	3	8275
All Departments	SALESMAN	4	5600
All Departments	PRESIDENT	1	5000
SALES	All Jobs	6	9400
SALES	CLERK	1	950
SALES	MANAGER	1	2850
SALES	SALESMAN	4	5600
RESEARCH	All Jobs	5	10875
RESEARCH	CLERK	2	1900
RESEARCH	ANALYST	2	6000
RESEARCH	MANAGER	1	2975
ACCOUNTING	All Jobs	3	8750
ACCOUNTING	CLERK	1	1300
ACCOUNTING	MANAGER	1	2450
ACCOUNTING	PRESIDENT	1	5000

18개의 행이 선택되었다.

step 4의 결과에 5건이 더 추가됨.

□ UNION ALL을 사용한 동등한 질의

– EMP, DEPT 테이블을 4번이나 반복 엑세스해야 함.

```
SELECT   DNAME, JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY DNAME, JOB
UNION ALL
SELECT   DNAME, 'All Jobs', COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY DNAME
UNION ALL
SELECT   'All Departments', JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY JOB
UNION ALL
SELECT   'All Departments', 'All Jobs', COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO ;
```

3. GROUPING SETS 함수

- grouping column list 의 각 경우에 대해 소계(subtotal)를 계산.
  - 함수의 인수로 grouping column list를 가짐.
  - Grouping column list 에 소계에 필요한 컬럼을 나열함.
  - Grouping colum의 수가 N이면, N level의 subtotal이 생성됨.

- 주의
  - CUBE의 인수는 계층 구조가 아님. 즉, 인수의 순서가 바뀌는 경우, 투플 간에 정렬 순서는 바뀔 수 있어도 데이터 결과는 같음.
  - 결과에 대한 정렬이 필요한 경우는 ORDER BY 절에 정렬 컬럼을 명시해야 함.

예제

ㅁ Step 1: 부서별, JOB별 인원수와 급여 합을 구하라.

```
SELECT   DNAME, 'All Jobs' JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY DNAME
UNION ALL
SELECT   'All Departments' DNAME, JOB, COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY JOB ;
```

DNAME	JOB	Total Empl	Total Sal
ACCOUNTING	All Jobs	3	8750
RESEARCH	All Jobs	5	10875
SALES	All Jobs	6	9400
All Departments	CLERK	4	4150
All Departments	SALESMAN	4	5600
All Departments	PRESIDENT	1	5000
All Departments	MANAGER	3	8275
All Departments	ANALYST	2	6000

8개의 행이 선택되었다.

□ Step 2 (Oracle): GROUPING SETS 함수 적용

```
SELECT   DECODE(GROUPING(DNAME), 1, 'All Departments', DNAME)
         AS DNAME,
         DECODE(GROUPING(JOB), 1, 'All Jobs', JOB) AS JOB,
         COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY GROUPING SETS (DNAME, JOB);
```

DNAME	JOB	Total Empl	Total Sal
All Departments	CLERK	4	4150
All Departments	SALESMAN	4	5600
All Departments	PRESIDENT	1	5000
All Departments	MANAGER	3	8275
All Departments	ANALYST	2	6000
ACCOUNTING	All Jobs	3	8750
RESEARCH	All Jobs	5	10875
SALES	All Jobs	6	9400

8개의 행이 선택되었다.

◻ Step 3 (Oracle): GROUPING SETS 함수 적용, 인자의 순서 변경

```
SELECT   DECODE(GROUPING(DNAME), 1, 'All Departments', DNAME)
         AS DNAME,
         DECODE(GROUPING(JOB), 1, 'All Jobs', JOB) AS JOB,
         COUNT(*) "Total Empl", SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY GROUPING SETS (JOB, DNAME);
```

DNAME	JOB	Total Empl	Total Sal
All Departments	CLERK	4	4150
All Departments	SALESMAN	4	5600
All Departments	PRESIDENT	1	5000
All Departments	MANAGER	3	8275
All Departments	ANALYST	2	6000
ACCOUNTING	All Jobs	3	8750
RESEARCH	All Jobs	5	10875
SALES	All Jobs	6	9400

8개의 행이 선택되었다.

결과는 step 2와 동일

□ ☞ 3개의 인수를 사용한 GROUPING SETS 함수 적용

```
SELECT   DNAME, JOB, MGR, SUM(SAL) "Total Sal"
FROM     EMP, DEPT
WHERE    DEPT.DEPTNO = EMP.DEPTNO
GROUP    BY GROUPING SETS ((DNAME, JOB, MGR), (DNAME, JOB), (JOB, MGR));
```

DNAME	JOB	MGR	Total Sal
SALES	CLERK	7698	950
ACCOUNTING	CLERK	7782	1300
RESEARCH	CLERK	7788	1100
RESEARCH	CLERK	7902	800
RESEARCH	ANALYST	7566	6000
SALES	MANAGER	7839	2850
RESEARCH	MANAGER	7839	2975
ACCOUNTING	MANAGER	7839	2450
SALES	SALESMAN	7698	5600
ACCOUNTING	PRESIDENT		5000
	CLERK	7698	950
	CLERK	7782	1300
	CLERK	7788	1100
	CLERK	7902	800
	ANALYST	7566	6000
	MANAGER	7839	8275
	SALESMAN	7698	5600
	PRESIDENT		5000
SALES	MANAGER		2850
SALES	CLERK		950
ACCOUNTING	CLERK		1300
ACCOUNTING	MANAGER		2450
ACCOUNTING	PRESIDENT		5000
RESEARCH	MANAGER		2975
SALES	SALESMAN		5600
RESEARCH	ANALYST		6000
RESEARCH	CLERK		1900

27개의 행이 선택되었다.

첫 번째 10건의 데이터는 (DNAME+JOB+MGR) 기준의 집계.
두 번째 8건의 데이터는 (JOB+MGR) 기준의 집계이며,
세 번째 9건의 데이터는 (DNAME+JOB) 기준의 집계임.

ANSI/ISO SQL

SQL DML : WINDOW FUNCTIONS

Window Functions의 종류

- 1. 그룹 내 순위(RANK) 관련 함수
  - RANK, DENSE_RANK, ROW_NUMBER 함수
  - 대부분의 DBMS에서 지원.
- 2. 그룹 내 집계(AGGREGATE) 관련 함수
  - SUM, MAX, MIN, AVG, COUNT 함수가 있다.
  - 대부분의 DBMS에서 지원. 단
  - SQL Server의 경우, OVER 절 내의 ORDER BY 지원 안함.
- 3. 그룹 내 행 순서 관련 함수
  - FIRST_VALUE, LAST_VALUE, LAG, LEAD 함수
  - Oracle에서만 지원
- 4. 그룹 내 비율 관련 함수
  - RATIO_TO_REPORT 함수는 Oracle에서만 지원.
  - PERCENT_RANK, CUME_DIST 함수는 ANSI/ISO SQL 표준과 Oracle DBMS에서 지원
  - NTILE 함수는 표준이 아니나 Oracle, SQL Server에서 모두 지원.
- 5.선형 분석을 포함한 통계 분석 관련 함수

□ Syntax

```
SELECT    WINDOW_FUNCTION (ARGUMENTS)
          OVER ([PARTITION BY 칼럼명] [ORDER BY 절] [WINDOWING 절])
FROM      테이블명;

WINDOWING 절 ::=
1.BETWEEN 사용 타입
     ROWS | RANGE BETWEEN
     UNBOUNDED PRECEDING | CURRENT ROW | VALUE_EXPR PRECEDING AND
     UNBOUNDED FOLLOWING | CURRENT ROW | VALUE_EXPR FOLLOWING
2.BETWEEN 미사용 타입
     ROWS | RANGE
     UNBOUNDED PRECEDING | CURRENT ROW | VALUE_EXPR PRECEDING/FOLLOWING
```

- PARTITION BY 절 : 전체 집합을 컬럼 기준에 의해 소그룹으로 나눔.
- ORDER BY 절 : 어떤 항목에 대해 순위를 지정할 지를 기술.
- WINDOWING 절 : 함수의 대상이 되는 투플의 범위를 강력하게 지정.
  - ◆ ROWS는 결과 투플의 수.
  - ◆ RANGE는 값에 의한 범위를 나타냄.
  - ◆ 둘 중의 하나를 선택해서 사용함.
  - ◆ SQL Server에서는 지원하지 않음.

1. 그룹 내 순위 관련 함수

◻ RANK 함수

– 예제: 급여가 높은 순서, 그리고 업무별 급여가 높은 순서를 구함.
단, 급여가 같으면 동일 순서로 처리함.

```
SELECT   JOB, ENAME, SAL,
         RANK( ) OVER (ORDER BY SAL DESC) ALL_RANK,
         RANK( ) OVER (PARTITION BY JOB ORDER BY SAL DESC) JOB_RANK
FROM     EMP;
```

JOB	ENAME	SAL	ALL_RANK	JOB_RANK
PRESIDENT	KING	5000	1	1
ANALYST	FORD	3000	2	1
ANALYST	SCOTT	3000	2	1
MANAGER	JONES	2975	4	1
MANAGER	BLAKE	2850	5	2
MANAGER	CLARK	2450	6	3
SALESMAN	ALLEN	1600	7	1
SALESMAN	TURNER	1500	8	2
CLERK	MILLER	1300	9	1
SALESMAN	WARD	1250	10	3
SALESMAN	MARTIN	1250	10	3
CLERK	ADAMS	1100	12	2
CLERK	JAMES	950	13	3
CLERK	SMITH	800	14	4

14개의 행이 선택되었다.

– 예제: 업무별 급여가 높은 순서를 구함. (업무별로 정렬함.)
단, 급여가 같으면 동일 순서로 처리함.

```
SELECT   JOB, ENAME, SAL,
         RANK( ) OVER (PARTITION BY JOB ORDER BY SAL DESC) JOB_RANK
FROM     EMP;
```

JOB	ENAME	SAL	JOB_RANK
ANALYST	FORD	3000	1
ANALYST	SCOTT	3000	1
CLERK	MILLER	1300	1
CLERK	ADAMS	1100	2
CLERK	JAMES	950	3
CLERK	SMITH	800	4
MANAGER	JONES	2975	1
MANAGER	BLAKE	2850	2
MANAGER	CLARK	2450	3
PRESIDENT	KING	5000	1
SALESMAN	ALLEN	1600	1
SALESMAN	TURNER	1500	2
SALESMAN	MARTIN	1250	3
SALESMAN	WARD	1250	3

◻ DENSE_RANK 함수

– 동일한 건수를 하나로 취급

```
SELECT   JOB, ENAME, SAL,
         RANK( ) OVER (ORDER BY SAL DESC) RANK,
         DENSE_RANK( ) OVER (ORDER BY SAL DESC) DENSE_RANK
FROM     EMP;
```

JOB	ENAME	SAL	RANK	DENSE_RANK
PRESIDENT	KING	5000	1	1
ANALYST	FORD	3000	2	2
ANALYST	SCOTT	3000	2	2
MANAGER	JONES	2975	4	3
MANAGER	BLAKE	2850	5	4
MANAGER	CLARK	2450	6	5
SALESMAN	ALLEN	1600	7	6
SALESMAN	TURNER	1500	8	7
CLERK	MILLER	1300	9	8
SALESMAN	WARD	1250	10	9
SALESMAN	MARTIN	1250	10	9
CLERK	ADAMS	1100	12	10
CLERK	JAMES	950	13	11
CLERK	SMITH	800	14	12

ROW_NUMBER 함수

– 동일한 값이라도 고유한 순위를 부여.

```
SELECT   JOB, ENAME, SAL,
         RANK( ) OVER (ORDER BY SAL DESC) RANK,
         ROW_NUMBER( ) OVER (ORDER BY SAL DESC) ROW_NUMBER
FROM     EMP;
```

JOB	ENAME	SAL	RANK	ROW_NUMBER
PRESIDENT	KING	5000	1	1
ANALYST	FORD	3000	2	2
ANALYST	SCOTT	3000	2	3
MANAGER	JONES	2975	4	4
MANAGER	BLAKE	2850	5	5
MANAGER	CLARK	2450	6	6
SALESMAN	ALLEN	1600	7	7
SALESMAN	TURNER	1500	8	8
CLERK	MILLER	1300	9	9
SALESMAN	WARD	1250	10	10
SALESMAN	MARTIN	1250	10	11
CLERK	ADAMS	1100	12	12
CLERK	JAMES	950	13	13
CLERK	SMITH	800	14	14

2. 일반 집계 관련 함수

ㅁ SUM 함수

– 예제: 같은 메니저를 갖는 모든 사원의 salary 합

```
SELECT   MGR, ENAME, SAL,
         SUM(SAL) OVER (PARTITION BY MGR) MGR_SUM
FROM     EMP;
```

MGR	ENAME	SAL	MGR_SUM
7566	FORD	3000	6000
7566	SCOTT	3000	6000
7698	JAMES	950	6550
7698	ALLEN	1600	6550
7698	WARD	1250	6550
7698	TURNER	1500	6550
7698	MARTIN	1250	6550
7782	MILLER	1300	1300
7788	ADAMS	1100	1100
7839	BLAKE	2850	8275
7839	JONES	2975	8275
7839	CLARK	2450	8275
7902	SMITH	800	800
	KING	5000	5000

– 예제: salary 합을 구하되, 조건은 이전 salary 까지의 누적값을 구함.

```
SELECT   MGR, ENAME, SAL,
         SUM(SAL) OVER (PARTITION BY MGR ORDER BY SAL
                        RANGE UNBOUNDED PRECEDING) as MGR_SUM
FROM     EMP;
```

MGR	ENAME	SAL	MGR_SUM
7566	SCOTT	3000	6000
7566	FORD	3000	6000
7698	JAMES	950	950
7698	WARD *	1250	3450
7698	MARTIN *	1250	3450
7698	TURNER	1500	4950
7698	ALLEN	1600	6550
7782	MILLER	1300	1300
7788	ADAMS	1100	1100
7839	CLARK	2450	2450
7839	BLAKE	2850	5300
7839	JONES	2975	8275
7902	SMITH	800	800
	KING	5000	5000

◻ MAX 함수

– 예제: 파티션 별로 최대값

```
SELECT   MGR, ENAME, SAL,
         MAX(SAL) OVER (PARTITION BY MGR) as MGR_MAX
FROM     EMP;
```

MGR	ENAME	SAL	MGR_MAX
7566	FORD	3000	3000
7566	SCOTT	3000	3000
7698	JAMES	950	1600
7698	ALLEN	1600	1600
7698	WARD	1250	1600
7698	TURNER	1500	1600
7698	MARTIN	1250	1600
7782	MILLER	1300	1300
7788	ADAMS	1100	1100
7839	BLAKE	2850	2975
7839	JONES	2975	2975
7839	CLARK	2450	2975
7902	SMITH	800	800
	KING	5000	5000

– 예제: 파티션 별로 최대값을 가진 투플만 추출

```
SELECT   MGR, ENAME, SAL
FROM     (SELECT   MGR, ENAME, SAL,
                   MAX(SAL) OVER (PARTITION BY MGR) as IV_MAX_SAL
          FROM     EMP)
WHERE    SAL = IV_MAX_SAL;
```

MGR	ENAME	SAL
7566	FORD	3000
7566	SCOTT	3000
7698	ALLEN	1600
7782	MILLER	1300
7788	ADAMS	1100
7839	JONES	2975
7902	SMITH	800
	KING	5000

◻ MIN 함수

```
SELECT   MGR, ENAME, HIREDATE, SAL,
         MIN(SAL) OVER (PARTITION BY MGR ORDER BY HIREDATE)
                  as MGR_MIN
FROM     EMP;
```

MGR	ENAME	HIREDATE	SAL	MGR_MIN
7566	FORD	1981-12-03	3000	3000
7566	SCOTT	1987-07-13	3000	3000
7698	ALLEN	1981-02-20	1600	1600
7698	WARD	1981-02-22	1250	1250
7698	TURNER	1981-09-08	1500	1250
7698	MARTIN	1981-09-28	1250	1250
7698	JAMES	1981-12-03	950	950
7782	MILLER	1982-01-23	1300	1300
7788	ADAMS	1987-07-13	1100	1100
7839	JONES	1981-04-02	2975	2975
7839	BLAKE	1981-05-01	2850	2850
7839	CLARK	1981-06-09	2450	2450
7902	SMITH	1980-12-17	800	800
	KING	1981-11-17	5000	5000

같은 파티션내에서
같은 날짜 별로
최소값을 구함.

◻ AVG 함수

– 평균을 구하되, 조건은 같은 매니저 내에서 자기 바로 앞과 바로 뒤의 직원만을 대상으로 함.

```
SELECT   MGR, ENAME, HIREDATE, SAL,
         ROUND (AVG(SAL) OVER (PARTITION BY MGR ORDER BY HIREDATE
         ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING)) as MGR_AVG
FROM     EMP;
```

MGR	ENAME	HIREDATE	SAL	MGR_AVG
7566	FORD	1981-12-03	3000	3000
7566	SCOTT	1987-07-13	3000	3000
7698	ALLEN	1981-02-20	1600	1425
7698	WARD	1981-02-22	1250	1450
7698	TURNER	1981-09-08	1500	1333
7698	MARTIN	1981-09-28	1250	1233
7698	JAMES	1981-12-03	950	1100
7782	MILLER	1982-01-23	1300	1300
7788	ADAMS	1987-07-13	1100	1100
7839	JONES	1981-04-02	2975	2913
7839	BLAKE	1981-05-01	2850	2758
7839	CLARK	1981-06-09	2450	2650
7902	SMITH	1980-12-17	800	800
	KING	1981-11-17	5000	5000

◻ COUNT 함수

– 예제: 사원들을 급여 기준으로 정렬하고, 각 사원에 대해 본인의 급여보다 50 적거나 150 많은 사이의 급여를 받는 직원수를 출력함.

```
SELECT   ENAME, SAL,
         COUNT(*) OVER (ORDER BY SAL
         RANGE BETWEEN 50 PRECEDING AND 150 FOLLOWING) as
    SIM_CNT
FROM     EMP;
```

ENAME	SAL	SIM_CNT	(범위값)
SMITH	800	2	(750~ 950)
JAMES	950	2	(900~1100)
ADAMS **	1100	3	(1050~1250)
WARD	1250	3	(1200~1400)
MARTIN	1250	3	(1200~1400)
MILLER	1300	3	(1250~1450)
TURNER	1500	2	(1450~1650)
ALLEN	1600	1	(1550~1750)
CLARK	2450	1	(2400~2600)
BLAKE	2850	4	(2800~3000)
JONES	2975	3	(2925~3125)
SCOTT	3000	3	(2950~3150)
FORD	3000	3	(2950~3150)
KING	5000	1	(4950~5150)

3. 그룹 내 행 순서 관련 함수

ㅁ FIRST_VALUE 함수

– 예제: 부서(파티션) 별로 직원들을 연봉이 높은 순서부터 정렬하고, 부서(파티션) 별로 가장 먼저 나온 값을 출력함

```
SELECT   DEPTNO, ENAME, SAL,
         FIRST_VALUE(ENAME) OVER (PARTITION BY DEPTNO ORDER BY
         SAL DESC ROWS UNBOUNDED PRECEDING) as DEPT_RICH
FROM     EMP;
```

DEPTNO	ENAME	SAL	DEPT_RICH
10	KING	5000	KING
10	CLARK	2450	KING
10	MILLER	1300	KING
20	SCOTT *	3000	SCOTT
20	FORD *	3000	SCOTT
20	JONES	2975	SCOTT
20	ADAMS	1100	SCOTT
20	SMITH	800	SCOTT
30	BLAKE	2850	BLAKE
30	ALLEN	1600	BLAKE
30	TURNER	1500	BLAKE
30	MARTIN	1250	BLAKE
30	WARD	1250	BLAKE
30	JAMES	950	BLAKE

FIRST_VALUE는 공동 등수를 인정하지 않고, 처음 나온 행만 처리함.

– 예제: ORDER BY를 사용하여 공동 등수 처리

```
SELECT    DEPTNO, ENAME, SAL,
          FIRST_VALUE(ENAME) OVER (PARTITION BY DEPTNO ORDER BY
    SAL DESC, ENAME ASC ROWS UNBOUNDED PRECEDING) as RICH_EMP
FROM      EMP;
```

DEPTNO	ENAME	SAL	RICH_EMP
10	KING	5000	KING
10	CLARK	2450	KING
10	MILLER	1300	KING
20	FORD	3000	FORD
20	SCOTT	3000	FORD
20	JONES	2975	FORD
20	ADAMS	1100	FORD
20	SMITH	800	FORD
30	BLAKE	2850	BLAKE
30	ALLEN	1600	BLAKE
30	TURNER	1500	BLAKE
30	MARTIN	1250	BLAKE
30	WARD	1250	BLAKE
30	JAMES	950	BLAKE

ㅁ LAST_VALUE 함수

– 예제: 파티션 별로 가장 마지막에 나온 값을 출력함.

```
SELECT   DEPTNO, ENAME, SAL,
         LAST_VALUE(ENAME) OVER (PARTITION BY DEPTNO ORDER BY SAL DESC
         ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING)
         AS DEPT_POOR
FROM     EMP;
```

DEPTNO	ENAME	SAL	DEPT_POOR
10	KING	5000	MILLER
10	CLARK	2450	MILLER
10	MILLER	1300	MILLER
20	SCOTT	3000	SMITH
20	FORD	3000	SMITH
20	JONES	2975	SMITH
20	ADAMS	1100	SMITH
20	SMITH	800	SMITH
30	BLAKE	2850	JAMES
30	ALLEN	1600	JAMES
30	TURNER	1500	JAMES
30	MARTIN	1250	JAMES
30	WARD	1250	JAMES
30	JAMES	950	JAMES

LAST_VALUE는 공동 등수를 인정하지 않고, 가장 나중에 나온 행만 처리함.

□ LAG 함수: 파티션별 윈도우에서 이전 몇 번째 행의 값을 가져옴.

– 예제: 직원들을 입사일자가 빠른 기준으로 정렬을 하고, 본인보다 입사일자가 한 명 앞선 사원의 급여를 본인의 급여와 함께 출력함.

```
SELECT   ENAME, HIREDATE, SAL,
         LAG(SAL) OVER (ORDER BY HIREDATE) AS PREV_SAL
FROM     EMP
WHERE    JOB = 'SALESMAN';
```

ENAME	HIREDATE	SAL	PREV_SAL
ALLEN	1981-02-20	1600	
WARD	1981-02-22	1250	1600
TURNER	1981-09-08	1500	1250
MARTIN	1981-09-28	1250	1500

4개의 행이 선택되었다.

– 예제: LAG 함수는 3개의 ARGUMENTS 까지 사용할 수 있는데, 두 번째 인자는 몇 번째 앞의 행을 가져올지 결정하는 것이고 (DEFAULT 1), 세 번째 인자는 NULL 인 경우 대체할 다른 값을 의미. 결과적으로 NVL이나 ISNULL 기능과 같음.

```
SELECT   ENAME, HIREDATE, SAL,
         LAG(SAL,2,0) OVER (ORDER BY HIREDATE) AS PREV_SAL
FROM     EMP
WHERE    JOB = 'SALESMAN';
```

ENAME	HIREDATE	SAL	PREV_SAL
ALLEN	1981-02-20	1600	0
WARD	1981-02-22	1250	0
TURNER	1981-09-08	1500	1600
MARTIN	1981-09-28	1250	1250

- LEAD 함수: 파티션별 윈도우에서 이후 몇 번째 행의 값을 가져옴.
  - 예제: 직원들을 입사일자가 빠른 기준으로 정렬을 하고, 바로 다음에 입사한 직원의 입사일자를 함께 출력함.

```
SELECT   ENAME, HIREDATE,
         LEAD(HIREDATE, 1) OVER (ORDER BY HIREDATE) AS "NEXTHIRED"
FROM     EMP;
```

ENAME	HIREDATE	NEXTHIRED
ALLEN	1981-02-20	1981-02-22
WARD	1981-02-22	1981-04-02
TURNER	1981-09-08	1981-09-28
MARTIN	1981-09-28	

4. 그룹 내 비율 관련 함수

◻ RATIO_TO_REPORT 함수

– 파티션 내 전체 SUM(칼럼) 값에 대한 투플별 컬럼 값의 백분율.

– 예제

```
SELECT    ENAME, SAL,
          ROUND(RATIO_TO_REPORT(SAL) OVER (), 2) AS R_R
FROM      EMP
WHERE     JOB = 'SALESMAN';
```

ENAME	SAL	R_R	
ALLEN	1600	0.29	(1600 / 5600)
WARD	1250	0.22	(1250 / 5600)
MARTIN	1250	0.22	(1250 / 5600)
TURNER	1500	0.27	(1500 / 5600)

4개의 행이 선택되었다.

◻ PERCENT_RANK 함수

– 파티션별 윈도우에서 제일 먼저 나오는 것을 0으로, 제일 늦게 나오는 것을 1로 하여, 값이 아닌 투플의 순서별 누적백분율.

– 예제

```
SELECT   DEPTNO, ENAME, SAL,
         PERCENT_RANK() OVER (PARTITION BY DEPTNO
                              ORDER BY SAL DESC) AS P_R
FROM     EMP;
```

DEPTNO	ENAME	SAL	P_R
10	KING	5000	0
10	CLARK	2450	0.5
10	MILLER	1300	1
20	SCOTT	3000	0
20	FORD	3000	0
20	JONES	2975	0.5
20	ADAMS	1100	0.75
20	SMITH	800	1
30	BLAKE	2850	0
30	ALLEN	1600	0.2
30	TURNER	1500	0.4
30	MARTIN	1250	0.6
30	WARD	1250	0.6
30	JAMES	950	1

14개의 행이 선택되었다.

ㅁ CUME_DIST 함수

– 파티션별 윈도우의 전체건수에서 현재 투플의 컬럼 값보다 작거나 같은 건수에 대한 누적백분율.

– 예제

```
SELECT   DEPTNO, ENAME, SAL,
         CUME_DIST() OVER (PARTITION BY DEPTNO
                           ORDER BY SAL DESC) AS CUME_DIST
FROM     EMP;
```

DEPTNO	ENAME	SAL	CUME_DIST
10	KING	5000	0.3333
10	CLARK	2450	0.6667
10	MILLER	1300	1.0000
20	SCOTT *	3000	0.4000
20	FORD *	3000	0.4000
20	JONES	2975	0.6000
20	ADAMS	1100	0.8000
20	SMITH	800	1.0000
30	BLAKE	2850	0.1667
30	ALLEN	1600	0.3333
30	TURNER	1500	0.5000
30	MARTIN **	1250	0.8333
30	WARD **	1250	0.8333
30	JAMES	950	1.0000

14개의 행이 선택되었다.

ㅁ NTILE 함수

– 파티션별 전체 건수를 ARGUMENT 값으로 N 등분한 결과.

– 예제: 전체 사원을 급여가 높은 순서로 정렬하고, 급여를 기준으로 4개의 그룹으로 분류함.

```
SELECT   DEPTNO, ENAME, SAL,
         NTILE(4) OVER (ORDER BY SAL DESC) AS QUAR_TILE
FROM     EMP;
```

DEPTNO	ENAME	SAL	QUAR_TILE
10	KING	5000	1
20	FORD	3000	1
20	SCOTT	3000	1
20	JONES	2975	1
30	BLAKE	2850	2
10	CLARK	2450	2
30	ALLEN	1600	2
30	TURNER	1500	2
10	MILLER	1300	3
30	WARD	1250	3
30	MARTIN	1250	3
20	ADAMS	1100	4
30	JAMES	950	4
20	SMITH	800	4

14개의 행이 선택되었다.