

## 2. 데이터 조회를 위한 고급 기법들

1. 집합 연산자들
2. 단일행 함수
3. 그룹함수



# 목차

- 기본적인 GROUP 함수
- GROUP BY절
- HAVING 절
- 자동 소계/합계 함수
- GROUPING 함수 살펴보기
- 그룹함수 연습문제
- 기타 그룹 함수

# 그룹함수의 종류

함수이름	의 미	사 용 예
COUNT	입력되는 데이터들의 건수를 출력	COUNT(sal)
SUM	입력되는 데이터들의 합계값을 출력	SUM(sal)
AVG	입력되는 데이터들의 평균값을 출력	AVG(sal)
MAX	입력되는 데이터들 중 최고 값을 출력	MAX(sal)
MIN	입력되는 데이터들 중 최저 값을 출력	MIN(sal)
STDDEV	입력되는 데이터 값들의 표준 편차값 출력	STDDEV(sal)
VARIANCE	입력되는 데이터 값들의 분산값 출력	VARIANCE(sal)
ROLLUP	입력되는 데이터들의 소계값을 자동으로 계산해서 출력	아래 예 참조
CUBE	입력되는 데이터들의 소계 및 전체 총계를 자동 계산 후 출력	아래 예 참조
GROUPING	해당 칼럼이 그룹에 사용되었는지 여부를 1 또는 0으로 반환	아래 예 참조
GROUPINGSET	한번의 질의로 여러개의 그룹화 가능	아래 예 참조
LISTAGG		아래 예 참조
PIVOT		아래 예 참조
LAG		아래 예 참조
LEAD		아래 예 참조
RANK		아래 예 참조
DENSE_RANK		아래 예 참조
누계집계하기		아래 예 참조

# 3.그룹함수- 기본 그룹 함수

## □ COUNT 함수

- ▣ 데이터의 총 건수를 반환

```
SELECT COUNT(*), COUNT(hpage) FROM professor;
```

## □ SUM 함수

- ▣ 입력된 데이터들의 합계 값을 구하는 함수

```
SELECT name, bonus FROM professor;
```

```
SELECT count(bonus), sum(bonus) FROM professor;
```

## □ AVG

- ▣ 입력된 값들의 평균을 구하는 함수

```
SELECT count(bonus), sum(bonus), avg(bonus), avg(NVL(bonus,0))  
FROM professor;
```

# 3.그룹함수- 기본 그룹 함수

## □ MAX/MIN

```
SELECT MAX(sal), MIN(sal) FROM emp;  
SELECT MAX(hiredate), MIN(hiredate) FROM emp;
```

-속도와 성능 부분에서 문제가 될 수 있으므로 인덱스를 활용하는 방법을  
사용 할 것을 적극 권장함

## □ STDDEV 함수/ VARIANCE 함수

```
SELECT STDDEV(pay), VARIANCE(pay) FROM professor;
```

### 3. 그룹함수- GROUP BY 절

#### □ GROUP BY

- ▣ 특정조건으로 세부적인 그룹화하기

- **Professor** 테이블에서 학과별 교수들의 평균 급여를 출력하세요.

```
SELECT deptno, AVG(NVL(pay,0)) "평균급여"  
FROM professor  
GROUP BY deptno;
```

- **Professor** 테이블에서 학과별, 직급별로 교수들의 평균 급여를 출력하세요.

```
SELECT deptno, position, AVG(NVL(pay,0)) "평균급여"  
FROM professor  
GROUP BY deptno, position;
```

## □ GROUP BY 절 사용 시 주의 사항

1. **SELECT 절에 사용된 그룹함수 이외의 칼럼이나 표현식은 반드시 GROUP BY 절에 사용되어야 합니다. 그렇지 않을 경우 아래와 같은 에러가 발생합니다.**

```
SELECT deptno, position, AVG(NVL(pay,0)) "평균급여"  
FROM professor GROUP BY deptno;
```

2. **GROUP BY 절에 사용된 칼럼은 SELECT 절에 사용되지 않아도 됩니다.**

```
SELECT deptno, AVG(NVL(pay,0)) "평균급여"  
FROM professor GROUP BY deptno, position;
```

3. **GROUP BY 절에는 반드시 칼럼 명이 사용되어야 하며 칼럼 Alias 는 사용하면 안됩니다.**

```
SELECT deptno "dno", AVG(NVL(pay,0)) "평균급여"  
FROM professor GROUP BY dno;
```


# HAVING 절

## □ HAVING

- GROUP BY 절에 조건을 지정할 때 사용
  - 평균 급여가 450 이상인 부서의 부서번호와 평균급여를 구하세요.

```
SELECT deptno, AVG(NVL(pay,0))  
FROM professor  
WHERE AVG(pay)>450  
GROUP BY deptno;
```

```
SELECT deptno, AVG(NVL(pay,0))  
FROM professor  
GROUP BY deptno  
HAVING AVG(pay)>450;
```





# 자동 소계/합계 함수

## □ ROLLUP 함수

- 자동으로 소계 값을 구해주는 함수

```
SELECT deptno, position,  
COUNT(*), SUM(pay)  
FROM professor  
GROUP BY ROLLUP(deptno, position);
```

이 부분들이  
ROLLUP에  
의해 자동으  
로  
구해진 소계  
부분 입니다.

101		3	1200
102	정 교수	1	490
102	조교수	1	350
102	전임강사	1	250
102		3	1090
103	정 교수	1	530
103	조교수	1	330
103	전임강사	1	290
103		3	1150
201	정 교수	1	570
201	조교수	1	330
201		2	900
202	조교수	1	310
202	전임강사	1	260
202		2	570
203	정교수	1	500
203		1	500
301	조교수	1	290
301	전임강사	1	220
301		2	510
		16	5920

## □ 실습

```
SELECT deptno, position, SUM(pay)
FROM professor
GROUP BY position, ROLLUP(deptno);
```

```
SELECT deptno, position, SUM(pay)
FROM professor
GROUP BY deptno, ROLLUP(position);
```

## □ CUBE 함수

- ROLLUP 함수와 같이 각 소계도 출력하고 전체 총계까지 출력.

```
SELECT deptno, position,  
count(*), SUM(pay)  
FROM professor  
GROUP BY CUBE(deptno, position);
```

이 부분이 ROLLUP  
함수와 다른 전체  
총계 출력부분입니  
다

	전임강사	5	1290
101		3	1200
101	정 교수	1	550
101	조교수	1	380
101	전임강사	1	270
102		3	1090
102	정 교수	1	490
102	조교수	1	350
102	전임강사	1	250
103		3	1150
103	정 교수	1	530
103	조교수	1	330
103	전임강사	1	290
201		2	900
201	정 교수	1	570
201	조교수	1	330
202		2	570
202	조교수	1	310
202	전임강사	1	260
203		1	500
203	정 교수	1	500
301		2	510
301	조교수	1	290
301	전임강사	1	220

## □ 실습 예:

```
SELECT deptno, position, SUM(pay)
FROM professor
GROUP BY deptno, CUBE(position);
```

```
SELECT deptno, position, SUM(pay)
FROM professor
GROUP BY position, CUBE(deptno);
```

# 다른 그룹핑 관련 함수들 살펴보기

## □ GROUPING 함수

- ▣ 그룹핑 작업에 사용 유무를 확인하는 함수

```
SELECT deptno, SUM(pay),  
GROUPING(deptno) g_deptno  
FROM professor  
GROUP BY ROLLUP(deptno);
```

DEPTNO	SUM(PAY)	G_DEPTNO
101	1200	0
102	1090	0
103	1150	0
201	900	0
202	570	0
203	500	0
301	510	0
	5920	1

8 rows selected.

부서별로 급여 합계를 구하는 쿼리에서 deptno 컬럼이 그룹핑 하는데 사용되었는지 살펴보기 위해 grouping 함수를 사용, 가장 마지막 합계부분만 1로 사용되지 않았고 나머지는 모두 0으로 사용되었음을 확인할 수 있음.

## □ 실습

```
SELECT deptno, position, SUM(pay),  
GROUPING(deptno) g_deptno,  
GROUPING(position) g_position  
FROM professor  
GROUP BY ROLLUP(deptno, position);
```

102	정교수	490	0	0
102	조교수	350	0	0
102	전임강사	250	0	0
102		1090	0	1
103	정교수	530	0	0
103	조교수	330	0	0
103	전임강사	290	0	0
103		1150	0	1
201	정교수	570	0	0
201	조교수	330	0	0
201		900	0	1
202	조교수	310	0	0
202	전임강사	260	0	0
202		570	0	1
203	정교수	500	0	0
203		500	0	1
301	조교수	290	0	0
301	전임강사	220	0	0
301		510	0	1
		5920	1	1

두 개의 컬럼을 그룹핑 하면서 각 컬럼의 사용 유무를 확인. G\_DEPTNO 컬럼은 모두 그룹핑하는데 사용되었고 G\_POSITION 컬럼은 각 부서별 소계 값을 구할 때는 그룹핑에 사용되지 않았음을 보여줌. 당연히 부서별 소계 값을 구하는 것이니 직급 컬럼은 사용되지 않을 것임

## □ GROUPING\_ID 함수

GROUPING 컬럼	BIT	GROUPING 결과	의 미
A , B	0 0	0	두 컬럼 다 GROUPING 에 사용됨
A	0 1	1	A 컬럼만 GROUPING 에 사용됨
B	1 0	2	B 컬럼만 GROUPING 에 사용됨
-	1 1	3	두 컬럼 모두 사용 안됨

## □ 실습 예

```
SELECT deptno, position, SUM(pay),
       GROUPING_ID(deptno, position) g_dp,
       GROUPING_ID(position, deptno) g_pd
FROM professor
GROUP BY ROLLUP(deptno, position);
```

101		1200	1	2
102	정 교수	490	0	0
102	조교수	350	0	0
102	전임강사	250	0	0
102		1090	1	2
103	정 교수	530	0	0
103	조교수	330	0	0
103	전임강사	290	0	0
103		1150	1	2
201	정 교수	570	0	0
201	조교수	330	0	0
201		900	1	2
202	조교수	310	0	0
202	전임강사	260	0	0
202		570	1	2
203	정 교수	500	0	0
203		500	1	2
301	조교수	290	0	0
301	전임강사	220	0	0

결과를 보면 GDP 부분에 0으로 되어 있는 건 두 컬럼 모두 그룹핑에 사용되었다는 뜻이고 1인 컬럼은 01 비트란 의미이므로 deptno는 그룹핑에 사용되었으나 position은 사용되지 않았다라는 의미, 이렇게 여러 개의 컬럼이 있을 경우 GROUPING\_ID를 활용하여 보다 간편하게 조회할 수 있음.



## □ GROUPING SETS

### 기존 방법

```
SELECT grade, COUNT(*)  
FROM student  
GROUP BY(grade)  
UNION  
SELECT deptno1, COUNT(*)  
FROM student  
GROUP BY(deptno1);
```

GRADE	COUNT(*)
1	5
2	5
3	5
4	5
101	4
102	4
103	2
201	6
202	2
301	2

10 rows selected.

SCOTT>

### GROUPING SETS 이용

```
SELECT grade, COUNT(*)  
FROM student  
GROUP BY GROUPING  
SETS(grade,deptno1);
```

GRADE	DEPTNO1	COUNT(*)
	102	4
	201	6
	301	2
	101	4
	202	2
	103	2
1		5
2		5
4		5
3		5

10 rows selected.

SCOTT>

## □ LISTAGG 함수(11g에서 추가됨)

문법 : LISTAGG(list\_column, ['구분자']) WITHIN GROUP(ORDER BY column)

```
SELECT deptno,  
       LISTAGG(name, '**') WITHIN GROUP(ORDER BY hiredate) "LISTAGG"  
FROM professor GROUP BY deptno;
```

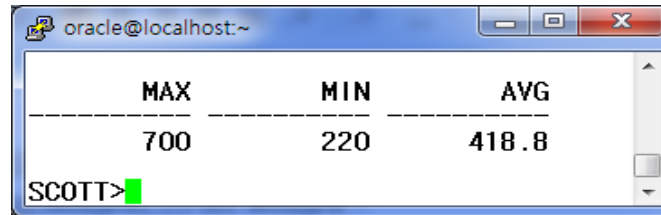
DEPTNO	LISTAGG
101	조인형**박승곤**송도권
102	주승재**김영조**양선희
103	김도형**나한열**김현정
201	심슨**최슬기
202	박원범**차범철
203	바비
301	허은**전민

rows selected.

```
SELECT deptno,  
       LISTAGG(name) WITHIN GROUP(ORDER BY hiredate) "LISTAGG"  
FROM professor GROUP BY deptno;
```

# SQL 그룹 함수 연습 문제

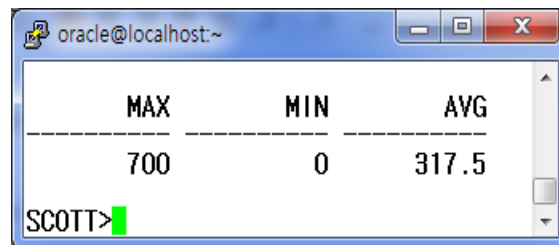
1) Professor 테이블을 사용하여 교수 중에서 급여(Pay)와 보너스(bonus)를 합친 금액이 가장 많은 경우와 가장 적은 경우, 평균 금액을 구하세요. 단 보너스가 없을 경우는 보너스를 0 으로 계산하고 출력 금액은 모두 소수점 첫째 자리까지만 나오게 하세요.



MAX	MIN	AVG
700	220	418.8

SCOTT>

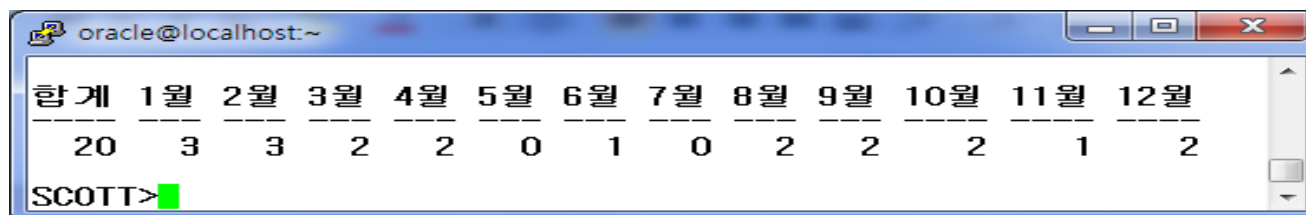
2) Professor 테이블을 사용하여 교수 중에서 급여(Pay)와 보너스(bonus)를 합친 금액이 가장 많은 경우와 가장 적은 경우, 평균 금액을 구하세요. 단 보너스가 없을 경우는 급여를 0 으로 계산하고 출력 금액은 모두 소수점 첫째 자리까지만 나오게 하세요.



MAX	MIN	AVG
700	0	317.5

SCOTT>

3) Student 테이블의 birthday 칼럼을 사용하여 아래 화면처럼 월별로 태어난 인원수를 출력하세요.

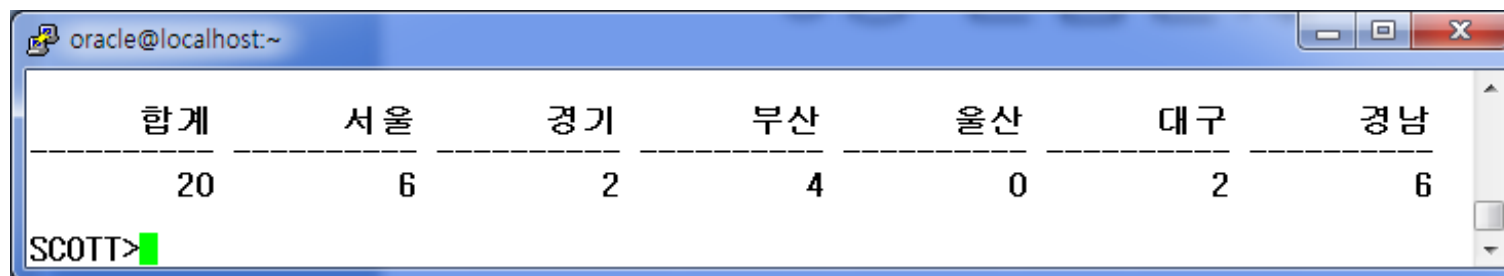


A terminal window titled 'oracle@localhost:~' showing the output of a SQL query. The output is a table with 13 columns: '합계' (Total), '1월' (Jan), '2월' (Feb), '3월' (Mar), '4월' (Apr), '5월' (May), '6월' (Jun), '7월' (Jul), '8월' (Aug), '9월' (Sep), '10월' (Oct), '11월' (Nov), and '12월' (Dec). The values are: 20, 3, 3, 2, 2, 0, 1, 0, 2, 2, 2, 1, 2. The prompt 'SCOTT>' is visible at the bottom left.

합계	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월	11월	12월
20	3	3	2	2	0	1	0	2	2	2	1	2

SCOTT>

4) Student 테이블의 tel 칼럼을 참고하여 아래와 같이 지역별 인원수를 출력하세요. 단 02 -서울 , 031 - 경기 , 051 - 부산 , 052 - 울산 , 053 - 대구 , 055 - 경남으로 출력하세요



A terminal window titled 'oracle@localhost:~' showing the output of a SQL query. The output is a table with 7 columns: '합계' (Total), '서울' (Seoul), '경기' (Gyeonggi), '부산' (Busan), '울산' (Ulsan), '대구' (Daegu), and '경남' (Gyeongnam). The values are: 20, 6, 2, 4, 0, 2, 6. The prompt 'SCOTT>' is visible at the bottom left.

합계	서울	경기	부산	울산	대구	경남
20	6	2	4	0	2	6

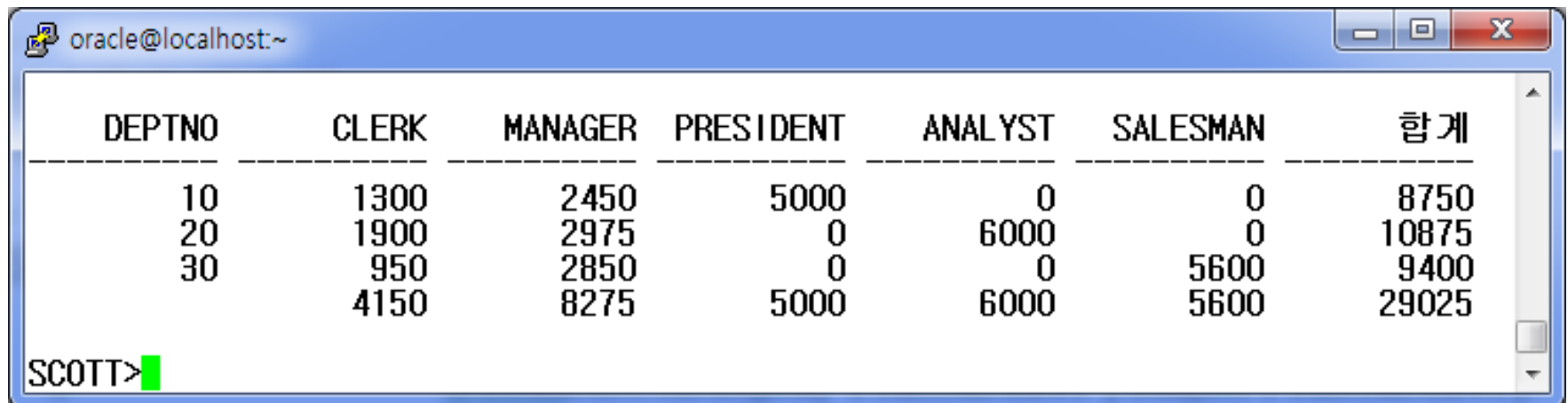
SCOTT>

5) Emp 테이블을 사용하여 아래의 화면과 같이 부서별로 직급별로 급여 합계 결과를 출력하세요. 먼저 아래의 두 건의 데이터를 입력 하신 후 작업하세요.

```
SQL>insert into emp (empno , deptno , ename , sal)
2 values (1000,10,'홍길동',3600) ;
```

```
SQL> insert into emp (empno , deptno , ename , sal)
2 values (2000,30,'일지매',3000);
```

```
SQL> commit;
```



The screenshot shows a terminal window titled 'oracle@localhost:~'. It displays a summary of employee salaries grouped by department (DEPTNO) and job grade (CLERK, MANAGER, PRESIDENT, ANALYST, SALESMAN). The '합계' (Total) column shows the sum of salaries for each department. The data is as follows:

DEPTNO	CLERK	MANAGER	PRESIDENT	ANALYST	SALESMAN	합계
10	1300	2450	5000	0	0	8750
20	1900	2975	0	6000	0	10875
30	950	2850	0	0	5600	9400
	4150	8275	5000	6000	5600	29025

The prompt 'SCOTT>' is visible at the bottom left of the window.

# 기타 함수

## LAG 함수 :

- 이전 행 값을 가져 올 때 사용하는 함수.

문 법 : LAG(출력할 컬럼명 , OFFSET , 기본 출력값)  
OVER (Query\_partition구문 , ORDER BY 정렬할 컬럼)

```
SCOTT>SELECT name, hiredate, pay,  
2          LAG(pay, 1, 0) OVER(ORDER BY hiredate) "LAG"  
3 FROM professor ;
```

NAME	HIREDATE	PAY	LAG
조인형	23-JUN-80	550	0
심슨	23-OCT-81	570	550
김도형	23-OCT-81	530	570
주승재	29-APR-82	490	530
바비	18-SEP-85	500	490
김영조	30-NOV-85	350	500
박승곤	30-JAN-87	380	350
나한열	01-JUL-97	330	380
송도권	22-MAR-98	270	330
박원범	01-DEC-99	310	270
허은	23-MAY-01	290	310
양선희	01-SEP-01	250	290
김현정	24-FEB-02	290	250
차범철	28-JAN-09	260	290
최슬기	30-AUG-09	330	260
전민	28-JUN-10	220	330

16 rows selected.

SCOTT>

```
SCOTT>SELECT name, hiredate, pay,  
2          LAG(pay, 3, 2) OVER(ORDER BY hiredate) "LAG"  
3 FROM professor ;
```

NAME	HIREDATE	PAY	LAG
조인형	23-JUN-80	550	2
심슨	23-OCT-81	570	2
김도형	23-OCT-81	530	2
주승재	29-APR-82	490	550
바비	18-SEP-85	500	570
김영조	30-NOV-85	350	530
박승곤	30-JAN-87	380	490
나한열	01-JUL-97	330	500
송도권	22-MAR-98	270	350
박원범	01-DEC-99	310	380
허은	23-MAY-01	290	330
양선희	01-SEP-01	250	270
김현정	24-FEB-02	290	310
차범철	28-JAN-09	260	290
최슬기	30-AUG-09	330	250
전민	28-JUN-10	220	290

16 rows selected.

SCOTT>

## □ ) LEAD 함수

- LEAD 함수는 LAG 함수와 반대로 이후의 값을 가져오는 함수.

oracle@localhost:~

```
SCOTT>SELECT name,hiredate,pay,  
2          LEAD(pay,1,0) OVER(ORDER BY hiredate) "LEAD"  
3 FROM professor ;
```

NAME	HIREDATE	PAY	LEAD
조인형	23-JUN-80	550	570
심스도	23-OCT-81	570	530
김도형	23-OCT-81	530	490
주승재	29-APR-82	490	500
바비	18-SEP-85	500	350
김영조	30-NOV-85	350	380
박승곤	30-JAN-87	380	330
나한열	01-JUL-97	330	270
송도권	22-MAR-98	270	310
박원범	01-DEC-99	310	290
허은진	23-MAY-01	290	250
양희선	01-SEP-01	250	290
김현정	24-FEB-02	290	260
차범철	28-JAN-09	260	330
최슬기	30-AUG-09	330	220
전민	28-JUN-10	220	0

16 rows selected.

SCOTT>

oracle@localhost:~

```
SCOTT>SELECT name,hiredate,pay,  
2          LEAD(pay,3,2) OVER(ORDER BY hiredate) "LEAD"  
3 FROM professor ;
```

NAME	HIREDATE	PAY	LEAD
조인형	23-JUN-80	550	490
심스도	23-OCT-81	570	500
김도형	23-OCT-81	530	350
주승재	29-APR-82	490	380
바비	18-SEP-85	500	330
김영조	30-NOV-85	350	270
박승곤	30-JAN-87	380	310
나한열	01-JUL-97	330	290
송도권	22-MAR-98	270	250
박원범	01-DEC-99	310	290
허은진	23-MAY-01	290	260
양희선	01-SEP-01	250	330
김현정	24-FEB-02	290	220
차범철	28-JAN-09	260	2
최슬기	30-AUG-09	330	2
전민	28-JUN-10	220	2

16 rows selected.

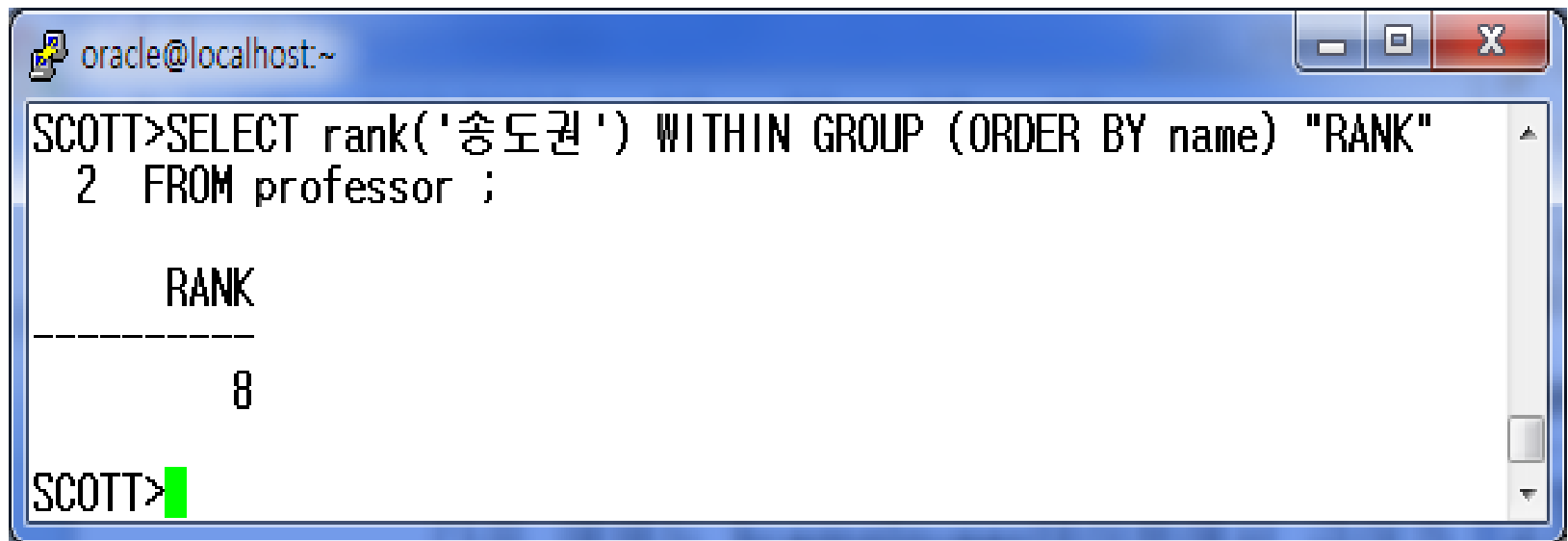
SCOTT>

## □ RANK 함수

### ▣ 순위 출력 함수

**RANK(조건값) WITHIN GROUP (ORDER BY 조건값 컬럼명 [ASC | DESC] )**

- 사용 예 : 이름이 '송도권' 인 교수의 순위를 조회하세요.



```
oracle@localhost:~  
SCOTT>SELECT rank('송도권') WITHIN GROUP (ORDER BY name) "RANK"  
2 FROM professor ;  
  
RANK  
-----  
8  
  
SCOTT>
```

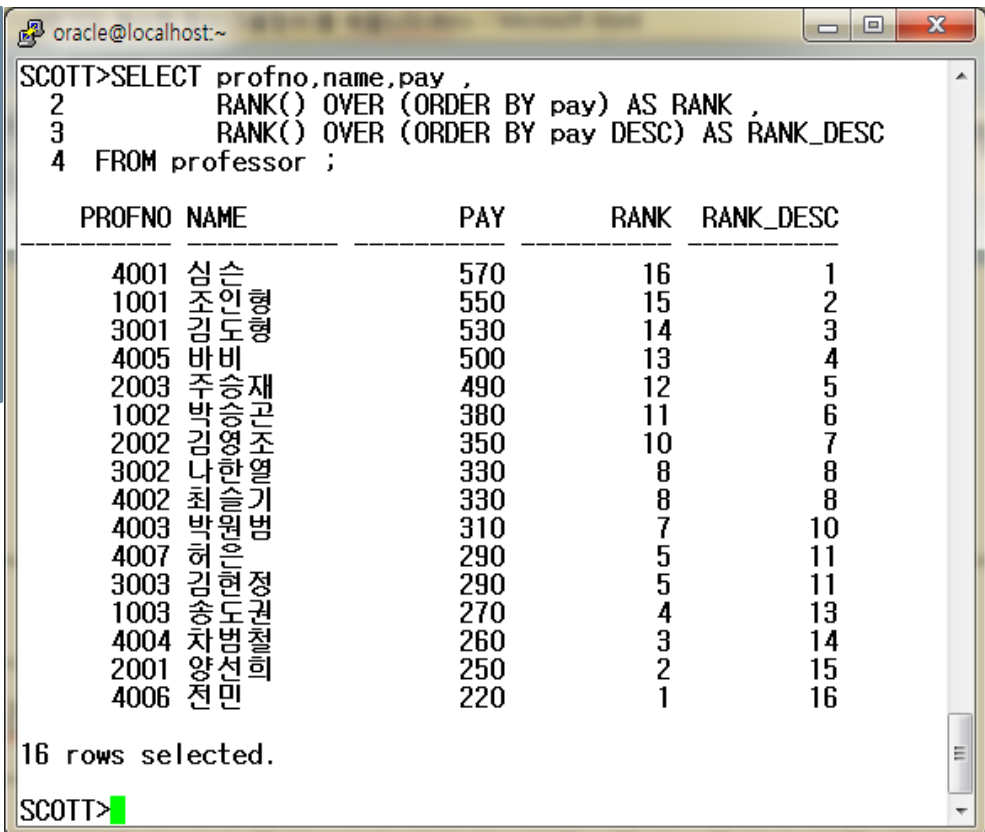
The screenshot shows a terminal window with the Oracle SQL prompt 'oracle@localhost:~'. The user enters the query 'SCOTT>SELECT rank('송도권') WITHIN GROUP (ORDER BY name) "RANK" 2 FROM professor ;'. The output displays the rank of the professor named '송도권' as 8. The prompt 'SCOTT>' is followed by a green cursor.



## □ RANK 함수 - 집계용

RANK ( ) (ORDER BY 조건컬럼명 [ASC | DESC] )

①교수 테이블  
( professor ) 테이블에서  
교수들의 교수번호와  
이름, 급여, 급여순위를  
출력하세요.



The screenshot shows an Oracle SQL\*Plus window with the title 'oracle@localhost:~'. The command prompt shows the following SQL query:

```
SCOTT>SELECT profno,name,pay ,  
2          RANK() OVER (ORDER BY pay) AS RANK ,  
3          RANK() OVER (ORDER BY pay DESC) AS RANK_DESC  
4 FROM professor ;
```

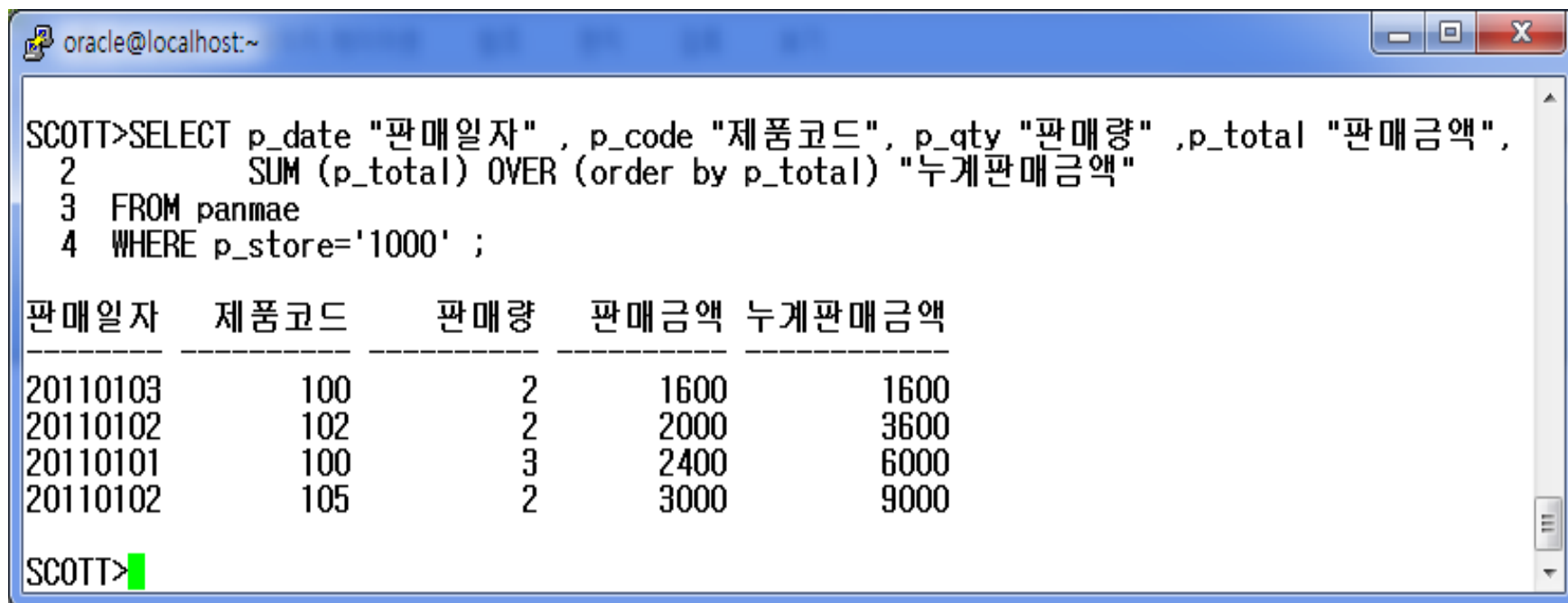
The query results are displayed in a table with the following columns: PROFNO, NAME, PAY, RANK, and RANK\_DESC. The results are sorted by PAY in descending order.

PROFNO	NAME	PAY	RANK	RANK_DESC
4001	심승민	570	16	1
1001	조인형	550	15	2
3001	김도형	530	14	3
4005	바비	500	13	4
2003	주승재	490	12	5
1002	박승곤	380	11	6
2002	김영조	350	10	7
3002	나한열	330	8	8
4002	최슬기	330	8	8
4003	박원범	310	7	10
4007	허은정	290	5	11
3003	김현정	290	5	11
1003	송도권	270	4	13
4004	차범철	260	3	14
2001	양선희	250	2	15
4006	전민	220	1	16

16 rows selected.  
SCOTT>

## □ 누적 합계 구하기

① panmae 테이블을 사용하여 1000 번 대리점의 판매 내역을 출력하되 판매 일자, 제품코드, 판매량, 누적 판매금액을 아래와 같이 출력하세요.



```
oracle@localhost:~  
SCOTT>SELECT p_date "판매일자" , p_code "제품코드", p_qty "판매량" ,p_total "판매금액",  
2          SUM (p_total) OVER (order by p_total) "누계판매금액"  
3 FROM panmae  
4 WHERE p_store='1000' ;
```

판매일자	제품코드	판매량	판매금액	누계판매금액
20110103	100	2	1600	1600
20110102	102	2	2000	3600
20110101	100	3	2400	6000
20110102	105	2	3000	9000

```
SCOTT>
```