

다양한 예제로 쉽게 배우는

오라클 SQL 과 PL/SQL

3장 SQL 복수행 함수 (그룹함수)를 배웁니다

3. SQL 복수행 함수

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. SQL 복수행 함수

1) COUNT 함수

- 입력되는 데이터의 총 건수를 반환합니다.

```
oracle@localhost:~
SCOTT>SELECT COUNT(*) , COUNT(hpage)
2 FROM professor;

COUNT(*) COUNT(HPAGE)
-----
16          4

SCOTT>
```

COUNT(*) 의 결과는 Null 값을 포함한 결과이고 COUNT(hpage) 의 결과는 Null 값을 제외한 결과입니다.

3. SQL 복수행 함수

2) SUM 함수

- 입력된 데이터들의 합계값을 구하는 함수입니다.

```
oracle@localhost:~
SCOTT>SET null (null);
SCOTT>SELECT name , bonus
2 FROM professor ;
```

NAME	BONUS
조인형	100
박승도	60
송도권	(null)
양선희	(null)
김영조	80
주승재	90
김도한	110
나형열	50
김현성	(null)
심슬기	130
최원범	(null)
박기철	50
차범철	(null)
바비	80
전민이	(null)
허영	30

```
16 rows selected.
SCOTT>
```

```
oracle@localhost:~
SCOTT>
SCOTT>SELECT COUNT(bonus) , SUM(bonus)
2 FROM professor ;
```

COUNT(BONUS)	SUM(BONUS)
10	780

```
SCOTT>
```

왼쪽 화면 보면 bonus 값을 가진 교수가 총 10명임을 알 수 있습니다. 그리고 그 합계가 780 인데 오른 쪽 위 화면에서 조회한 내용과 동일함을 알 수 있습니다.

3. SQL 복수행 함수

3) AVG 함수

- 입력된 값들의 평균값을 구해주는 함수입니다.

```
oracle@localhost:~
SCOTT>SELECT COUNT(bonus),SUM(bonus),AVG(bonus)
2 FROM professor ;

COUNT(BONUS) SUM(BONUS) AVG(BONUS)
-----
10          780          78

SCOTT>
```

NULL 값 자동 제외. 틀린 결과

```
oracle@localhost:~
SCOTT>SELECT COUNT(*),SUM(bonus),
2          AVG(NVL(bonus,0))
3 FROM professor ;

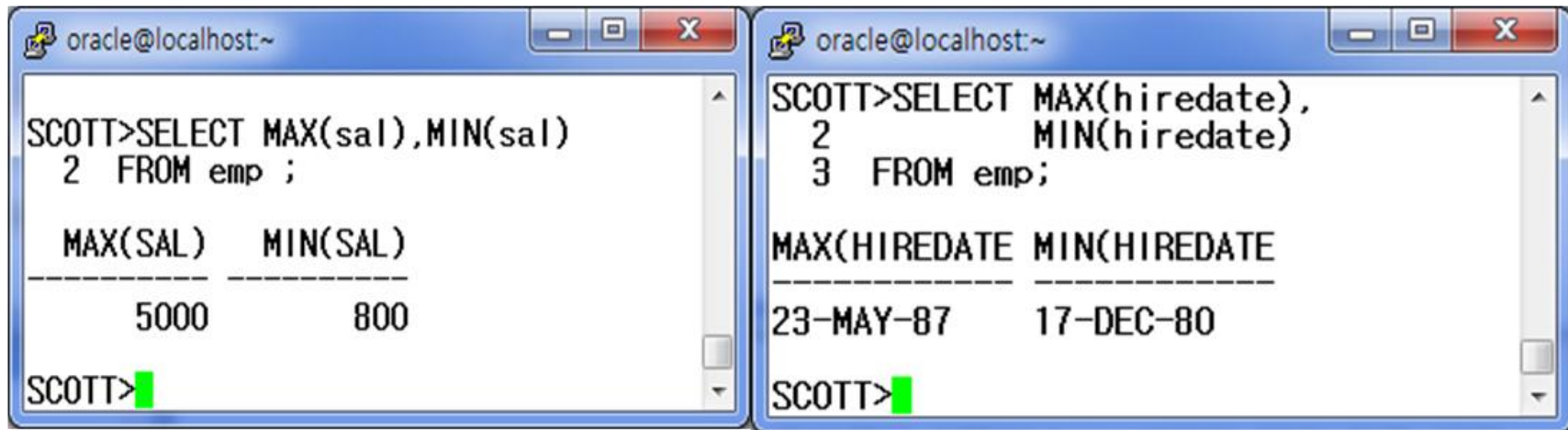
COUNT(*) SUM(BONUS) AVG(NVL(BONUS,0))
-----
16          780          48.75

SCOTT>
```

NULL 값을 0으로 변환.
바른 결과

3. SQL 복수행 함수

4) MAX 함수 / MIN 함수



The image shows two side-by-side screenshots of an Oracle SQL*Plus terminal window. The left window shows the execution of a query to find the maximum and minimum salary in the 'emp' table. The right window shows the execution of a query to find the maximum and minimum hire date in the 'emp' table.

Left Window:

```
SCOTT>SELECT MAX(sal),MIN(sal)
2 FROM emp ;
```

MAX(SAL)	MIN(SAL)
5000	800

SCOTT>

Right Window:

```
SCOTT>SELECT MAX(hiredate),
2 MIN(hiredate)
3 FROM emp;
```

MAX(HIREDATE)	MIN(HIREDATE)
23-MAY-87	17-DEC-80

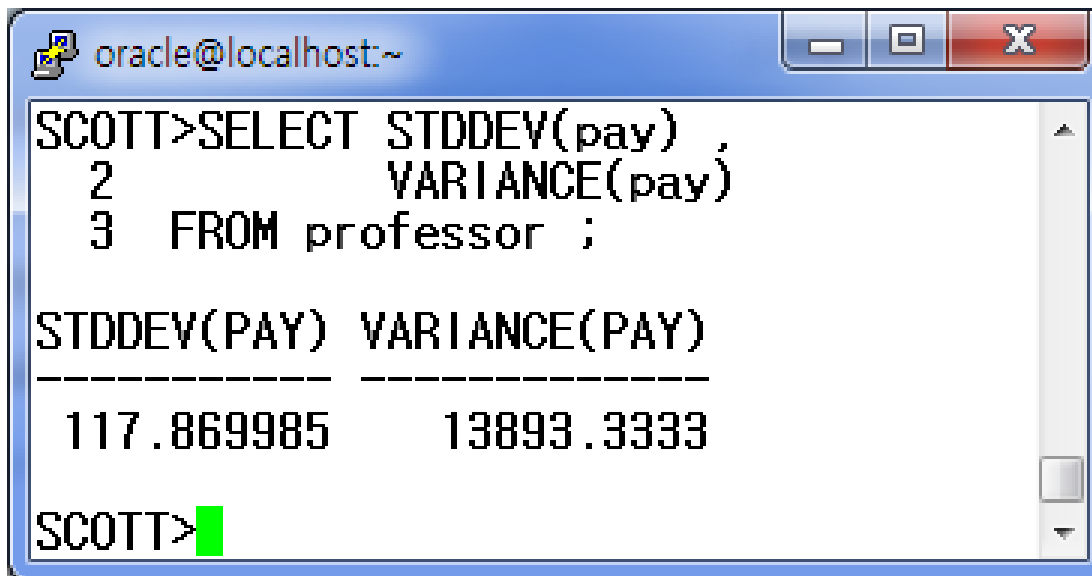
SCOTT>

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

3. SQL 복수행 함수

5) STDDEV 함수 / VARIANCE 함수

- STDDEV 함수는 표준편차를 구하는 함수이고 VARIANCE 함수는 분산을 구하는 함수입니다.



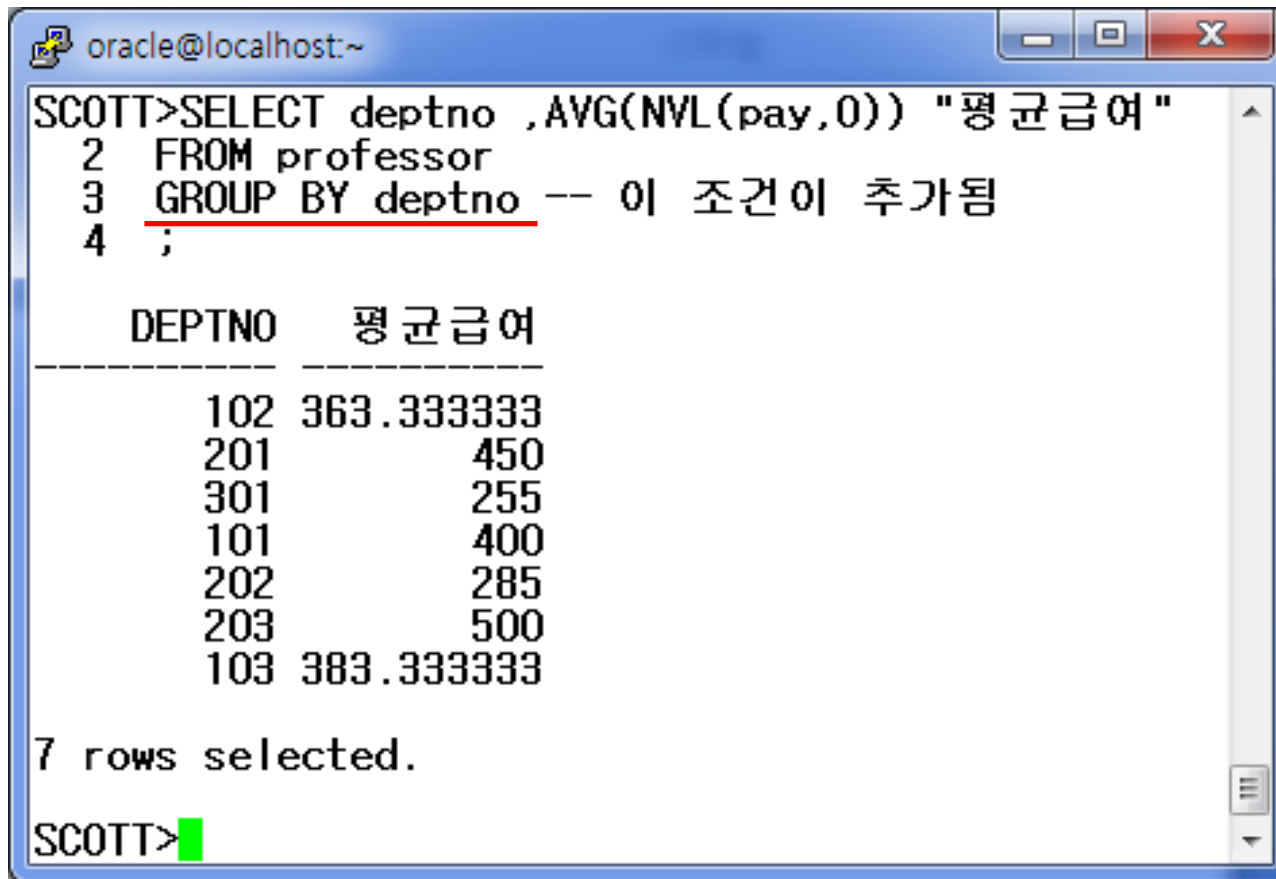
```
oracle@localhost:~  
SCOTT>SELECT STDDEV(pay) ,  
2          VARIANCE(pay)  
3  FROM professor ;  
  
STDDEV(PAY)  VARIANCE(PAY)  
-----  
117.869985   13893.3333  
  
SCOTT>
```


3. SQL 복수행 함수

2. 특정 조건으로 세부적인 그룹화 하기
(GROUP BY 절 사용하기)

3. SQL 복수행 함수

- Professor 테이블에서 학과별로 교수들의 평균 급여를 출력하세요.



```
oracle@localhost:~  
SCOTT>SELECT deptno ,AVG(NVL(pay,0)) "평균급여"  
2 FROM professor  
3 GROUP BY deptno -- 이 조건이 추가됨  
4 ;  
  
DEPTNO    평균급여  
-----  
102      363.333333  
201              450  
301              255  
101              400  
202              285  
203              500  
103      383.333333  
  
7 rows selected.  
SCOTT>
```

3. SQL 복수행 함수

- Professor 테이블에서 학과별, 직급별로 교수들의 평균 급여를 출력하세요.

```

oracle@localhost:~
SCOTT>set pagesize 50
SCOTT>SELECT deptno , position , AVG(NVL(pay,0)) "평균급여"
2 FROM professor
3 GROUP BY deptno, position -- position 조건 추가됨 ;

```

DEPTNO	POSITION	평균급여
201	조교수	330
202	조교수	310
301	전임강사	220
301	조교수	290
102	전임강사	250
103	정교수	530
103	전임강사	290
203	정교수	500
101	전임강사	270
202	전임강사	260
101	조교수	380
201	정교수	570
101	정교수	550
103	조교수	330
102	조교수	350
102	정교수	490

```

16 rows selected.
SCOTT>

```

3. SQL 복수행 함수

- 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.

```

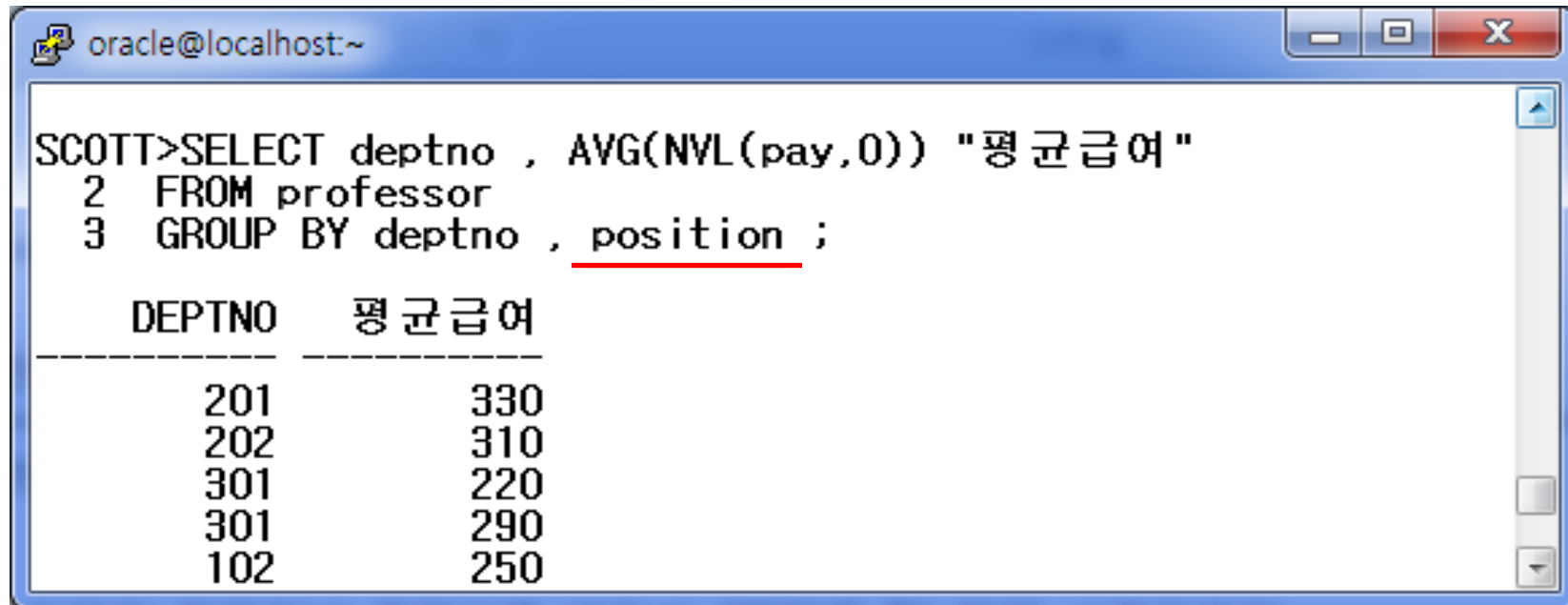
oracle@localhost:~
SCOTT>SELECT deptno , position , AVG(NVL(pay,0)) "평균급여"
2   FROM professor
3   GROUP BY deptno -- position 조건이 누락되었습니다 ;
SELECT deptno , position , AVG(NVL(pay,0)) "평균급여"
          *
ERROR at line 1:
ORA-00979: not a GROUP BY expression

SCOTT>

```

3. SQL 복수행 함수

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



The screenshot shows a terminal window titled 'oracle@localhost:~'. The user 'SCOTT' has executed the following SQL query:

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

The query results are displayed in a table with two columns: 'DEPTNO' and '평균급여' (Average Salary). The results are as follows:

DEPTNO	평균급여
201	330
202	310
301	220
301	290
102	250

3. SQL 복수행 함수

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

```
oracle@localhost:~
SCOTT>SELECT deptno dno,AVG(NVL(pay,0)) "평균급여"
2 FROM professor
3 GROUP BY dno ;
GROUP BY dno
          *
ERROR at line 3:
ORA-00904: "DNO": invalid identifier

SCOTT>
```

3. SQL 복수행 함수

3. 조건을 주고 검색하기
(HAVING 절 사용하기)

3. SQL 복수행 함수

- 평균 급여가 450 이상인 부서의 부서번호와 평균급여를 구하세요.

```
oracle@localhost:~
SCOTT>SELECT deptno ,AVG(NVL(pay,0))
2 FROM professor
3 WHERE AVG(pay) > 450
4 GROUP BY deptno ;
WHERE AVG(pay) > 450
*
ERROR at line 3:
ORA-00934: group function is not allowed here

SCOTT>
```

```
oracle@localhost:~
SCOTT>SELECT deptno ,AVG(NVL(pay,0))
2 FROM professor
3 GROUP BY deptno
4 HAVING AVG(pay) > 450 ;

DEPTNO AVG(NVL(PAY,0))
-----
203          500

SCOTT>
```


3. SQL 복수행 함수

4. 자동으로 소계 / 합계를 구해주는 함수

3. SQL 복수행 함수

6) ROLLUP 함수

자동으로 소계 값을 구해주는 함수

이 부분들이
ROLLUP에
의해 자동으로
구해진 소계
부분 입니다.

oracle@localhost:~

```
SCOTT>SELECT deptno,position ,
2          COUNT(*) , SUM(pay)
3 FROM professor
4 GROUP BY ROLLUP(deptno,position);
```

DEPTNO	POSITION	COUNT(*)	SUM(PAY)
101	정 교수	1	550
101	조 교수	1	380
101	전임강사	1	270
101		3	1200
102	정 교수	1	490
102	조 교수	1	350
102	전임강사	1	250
102		3	1090
103	정 교수	1	530
103	조 교수	1	330
103	전임강사	1	290
103		3	1150
201	정 교수	1	570
201	조 교수	1	330
201		2	900
202	조 교수	1	310
202	전임강사	1	260
202		2	570
203	정 교수	1	500
203		1	500
301	조 교수	1	290
301	전임강사	1	220
301		2	510
		16	5920

24 rows selected.

SCOTT>

3. SQL 복수행 함수

```
oracle@localhost:~
SCOTT>set pagesize 50
SCOTT>SELECT deptno,position, SUM(pay)
2 FROM professor
3 GROUP BY position ,ROLLUP(deptno) ;
```

DEPTNO	POSITION	SUM(PAY)
101	정교수	550
102	정교수	490
103	정교수	530
201	정교수	570
203	정교수	500
		2640
101	조교수	380
102	조교수	350
103	조교수	330
201	조교수	330
202	조교수	310
301	조교수	290
		1990
101	전임강사	270
102	전임강사	250
103	전임강사	290
202	전임강사	260
301	전임강사	220
		1290

19 rows selected.

SCOTT>

```
oracle@localhost:~
SCOTT>SELECT deptno,position, SUM(pay)
2 FROM professor
3 GROUP BY deptno,ROLLUP(position) ;
```

DEPTNO	POSITION	SUM(PAY)
101	정교수	550
101	조교수	380
101	전임강사	270
101		1200
102	정교수	490
102	조교수	350
102	전임강사	250
102		1090
103	정교수	530
103	조교수	330
103	전임강사	290
103		1150
201	정교수	570
201	조교수	330
201		900
202	조교수	310
202	전임강사	260
202		570
203	정교수	500
203		500
301	조교수	290
301	전임강사	220
301		510

23 rows selected.

SCOTT>

3. SQL 복수행 함수

7) CUBE 함수

- ROLLUP 함수와
같이 각 소계도 출력
하고 전체 총계까지
출력합니다.

이 부분이 ROLLUP
함수와 다른 전체
총계 출력부분입니
다

oracle@localhost:~

```
SCOTT>SELECT deptno,position ,
2          COUNT(*) , SUM(pay)
3 FROM professor
4 GROUP BY CUBE(deptno,position);
```

DEPTNO	POSITION	COUNT(*)	SUM(PAY)
		16	5920
	정 교수	5	2640
	조 교수	6	1990
	전임강사	5	1290
101		3	1200
101	정 교수	1	550
101	조 교수	1	380
101	전임강사	1	270
102		3	1090
102	정 교수	1	490
102	조 교수	1	350
102	전임강사	1	250
103		3	1150
103	정 교수	1	530
103	조 교수	1	330
103	전임강사	1	290
201		2	900
201	정 교수	1	570
201	조 교수	1	330
202		2	570
202	조 교수	1	310
202	전임강사	1	260
203		1	500
203	정 교수	1	500
301		2	510
301	조 교수	1	290
301	전임강사	1	220

27 rows selected.

SCOTT>

3. SQL 복수행 함수

```
oracle@localhost:~
SCOTT>SELECT deptno,position, SUM(pay)
2 FROM professor
3 GROUP BY deptno,CUBE(position) ;
```

DEPTNO	POSITION	SUM(PAY)
101		1200
101	정교수	550
101	조교수	380
101	전임강사	270
102		1090
102	정교수	490
102	조교수	350
102	전임강사	250
103		1150
103	정교수	530
103	조교수	330
103	전임강사	290
201		900
201	정교수	570
201	조교수	330
202		570
202	조교수	310
202	전임강사	260
203		500
203	정교수	500
301		510
301	조교수	290
301	전임강사	220

23 rows selected.

SCOTT>

```
oracle@localhost:~
SCOTT>
SCOTT>SELECT deptno,position, SUM(pay)
2 FROM professor
3 GROUP BY position,CUBE(deptno) ;
```

DEPTNO	POSITION	SUM(PAY)
	정교수	2640
101	정교수	550
102	정교수	490
103	정교수	530
201	정교수	570
203	정교수	500
	조교수	1990
101	조교수	380
102	조교수	350
103	조교수	330
201	조교수	330
202	조교수	310
301	조교수	290
	전임강사	1290
101	전임강사	270
102	전임강사	250
103	전임강사	290
202	전임강사	260
301	전임강사	220

19 rows selected.

SCOTT>

3. SQL 복수행 함수

5. 다른 그룹핑 관련 함수들 살펴보기

3. SQL 복수행 함수

1) GROUPING 함수 - 그룹핑 작업에 사용 유무를 확인하는 함수

```
oracle@localhost:~  
SCOTT>SELECT deptno, SUM(pay),  
2          GROUPING(deptno) g_deptno  
3 FROM professor  
4 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.  
SCOTT>
```

이 예제는 부서별로 급여 합계를 구하는 쿼리입니다. 이 쿼리에서 deptno 컬럼이 그룹핑 하는데 사용되었는지 살펴 보기 위해 grouping 함수를 사용했는데 가장 마지막 합계부분만 1로 사용되지 않았고 나머지는 모두 0으로 사용되었음을 확인 할 수 있습니다.

3. SQL 복수행 함수

```

oracle@localhost:~
SCOTT>SELECT deptno,position,SUM(pay),
2          GROUPING(deptno) g_deptno ,
3          GROUPING(position) g_position
4 FROM professor
5 GROUP BY ROLLUP(deptno,position) ;

```

DEPTNO	POSITION	SUM(PAY)	G_DEPTNO	G_POSITION
101	정 교수	550	0	0
101	조교수	380	0	0
101	전임강사	270	0	0
101		1200	0	1
102	정 교수	490	0	0
102	조교수	350	0	0
102	전임강사	250	0	0
102		1090	0	1
103	정 교수	530	0	0
103	조교수	330	0	0
103	전임강사	290	0	0
103		1150	0	1
201	정 교수	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	정 교수	500	0	0
203		500	0	1
301	조교수	290	0	0
301	전임강사	220	0	0
301		510	0	1
		5920	1	1

24 rows selected.

SCOTT>

이 예는 두 개의 컬럼을 그룹핑 하면서 각 컬럼의 사용 유무를 확인했습니다.

G_DEPTNO 컬럼은 모두 그룹핑 하는데 사용되었고 G_POSITION 컬럼은 각 부서별 소계 값을 구할 때는 그룹핑에 사용되지 않았음을 보여 줍니다. 당연히 부서별 소계 값을 구하는 것이니 직급 컬럼은 사용되지 않을 것입니다.

3. SQL 복수행 함수

2) 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

3. SQL 복수행 함수

```

oracle@localhost:~
SCOTT>SELECT deptno,position,SUM(pay),
2          GROUPING_ID(deptno,position) GDP ,
3          GROUPING_ID(position,deptno) GPD
4 FROM professor
5 GROUP BY ROLLUP(deptno,position) ;

```

DEPTNO	POSITION	SUM(PAY)	GDP	GPD
101	정 교수	550	0	0
101	조 교수	380	0	0
101	전임강사	270	0	0
101		1200	1	2
102	정 교수	490	0	0
102	조 교수	350	0	0
102	전임강사	250	0	0
102		1090	1	2
103	정 교수	530	0	0
103	조 교수	330	0	0
103	전임강사	290	0	0
103		1150	1	2
201	정 교수	570	0	0
201	조 교수	330	0	0
201		900	1	2
202	조 교수	310	0	0
202	전임강사	260	0	0
202		570	1	2
203	정 교수	500	0	0
203		500	1	2
301	조 교수	290	0	0
301	전임강사	220	0	0
301		510	1	2
		5920	3	3

24 rows selected.

```

SCOTT>

```

위 화면을 보면 GDP 부분에 0으로 되어 있는 건 두 컬럼 모두 그룹핑에 사용되었다는 뜻이고 1인 컬럼은 01 비트란 의미이므로 deptno 는 그룹핑에 사용되었으나 position은 사용되지 않았다 라는 의미 입니다. 이렇게 여러 개의 컬럼이 있을 경우 GROUPING_ID 를 활용하여 보다 간편하게 조회 할 수 있습니다.

3. SQL 복수행 함수

3) GROUPING SETS

기존 방법

```
oracle@localhost:~
SCOTT>SELECT grade , COUNT(*)
2 FROM student
3 GROUP BY(grade)
4 UNION
5 SELECT deptno1, COUNT(*)
6 FROM student
7 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

```
oracle@localhost:~
SCOTT>SELECT grade,deptno1 , COUNT(*)
2 FROM student
3 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>
```

3. SQL 복수행 함수

4) LISTAGG 함수 (11g 에서 추가됨)

```
oracle@localhost:~  
SCOTT>SET LINE 200  
SCOTT>COL listagg FOR a40  
SCOTT>  
SCOTT>SELECT deptno ,  
2          LISTAGG(name,**') WITHIN GROUP(ORDER BY hiredate) "LISTAGG"  
3 FROM professor  
4 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  
  
7 rows selected.  
SCOTT>
```

3. SQL 복수행 함수

```

oracle@localhost:~
SCOTT>SELECT deptno ,
2          LISTAGG(name) WITHIN GROUP(ORDER BY hiredate) "LISTAGG"
3 FROM professor
4 GROUP BY deptno ;

DEPTNO LISTAGG
-----
101  조인형 박승곤 송도권
102  주승재 김영조 양선희
103  김도형 나한열 김현정
201  심슨최슬기
202  박원범 차범철
203  바비
301  허은전민

7 rows selected.

SCOTT>

```

3. SQL 복수행 함수

5) PIVOT 함수 (11g 버전에서 추가된 함수)

PIVOT 기능을 사용하지 않고 출력

```
SCOTT>col sun for a3
SCOTT>col mon for a3
SCOTT>col tue for a3
SCOTT>col wed for a3
SCOTT>col thu for a3
SCOTT>col fri for a3
SCOTT>col sat for a3
SCOTT>SELECT MAX(Decode(day, '일', num_day)) AS SUN,
2          MAX(Decode(day, '월', num_day)) AS MON,
3          MAX(Decode(day, '화', num_day)) AS TUE,
4          MAX(Decode(day, '수', num_day)) AS WED,
5          MAX(Decode(day, '목', num_day)) AS THU,
6          MAX(Decode(day, '금', num_day)) AS FRI,
7          MAX(Decode(day, '토', num_day)) AS SAT
8 FROM cal
9 GROUP BY week
10 ORDER BY week ;
```

SUN	MON	TUE	WED	THU	FRI	SAT
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

SCOTT>

3. SQL 복수행 함수

```

oracle@localhost:~
SCOTT>col "주" for a3
SCOTT>SELECT * FROM (SELECT week "주", day, num_day
2 FROM cal )
3 PIVOT
4 (
5 MAX(num_day)
6 FOR day IN('일' as "일",
7 '월' as "월",
8 '화' as "화",
9 '수' as "수",
10 '목' as "목",
11 '금' as "금",
12 '토' as "토")
13 )
14 ORDER BY "주" ;

주 일 월 화 수 목 금 토
--- -- -- -- -- -- --
1 1 2 3 4 5 6 7
2 8 9 10 11 12 13 14
3 15 16 17 18 19 20 21
4 22 23 24 25 26 27 28
5 29 30 31

SCOTT>

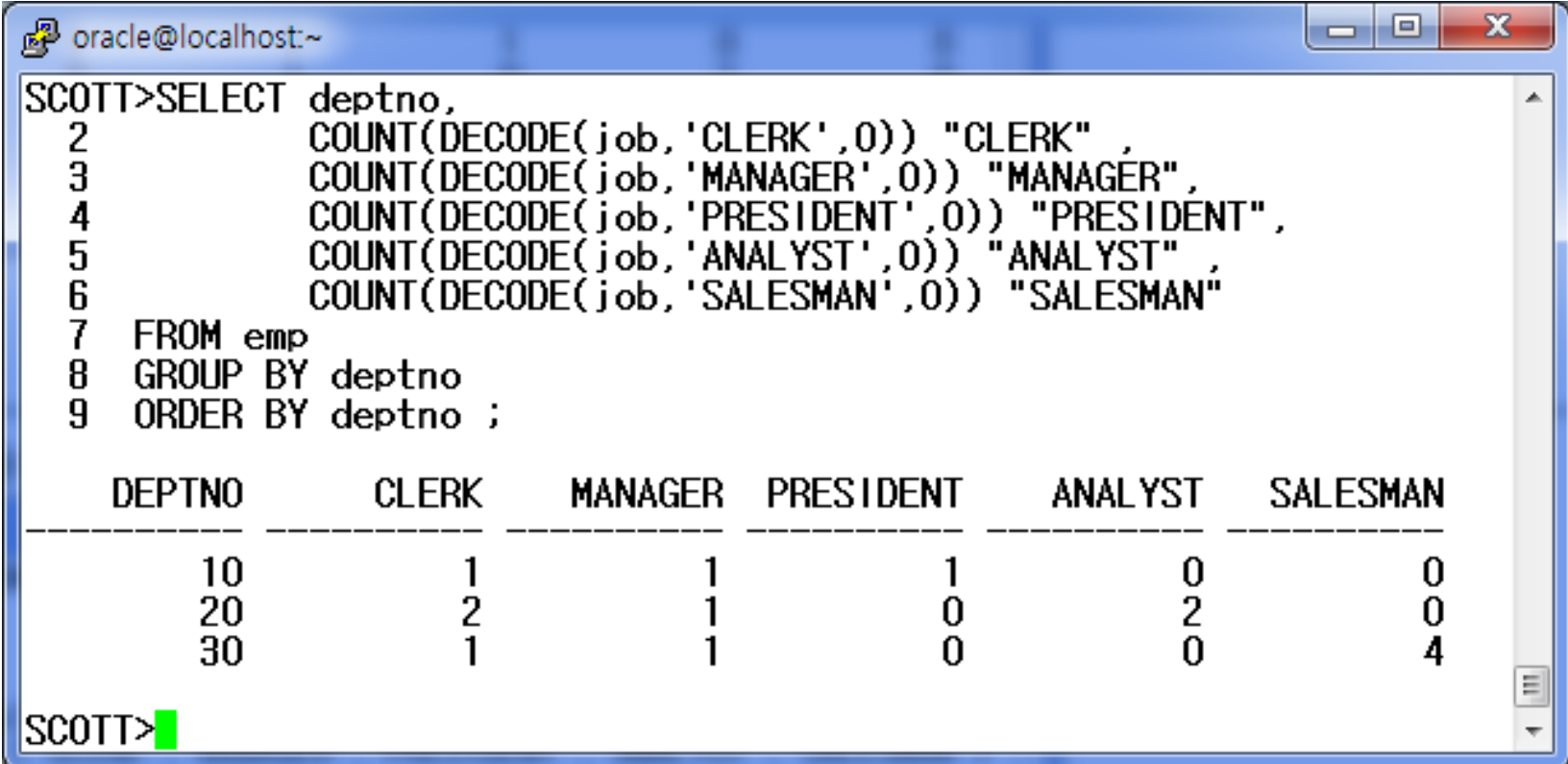
```

PIVOT 절에
MAX(num_day) 절은
DECODE 문장에서 사용
되는 함수를 적으면 되고
FOR 절에는 화면에 집계
될 그룹핑 할 칼럼을 적
으면 됩니다.

3. SQL 복수행 함수

③ EMP 테이블에서 부서별로 각 직급별 인원이 몇 명인 지 계산해서 출력하세요

- DECODE 함수를 이용하는 방법



```
oracle@localhost:~  
SCOTT>SELECT deptno,  
2          COUNT(DECODE(job, 'CLERK', 0)) "CLERK" ,  
3          COUNT(DECODE(job, 'MANAGER', 0)) "MANAGER",  
4          COUNT(DECODE(job, 'PRESIDENT', 0)) "PRESIDENT",  
5          COUNT(DECODE(job, 'ANALYST', 0)) "ANALYST" ,  
6          COUNT(DECODE(job, 'SALESMAN', 0)) "SALESMAN"  
7 FROM emp  
8 GROUP BY deptno  
9 ORDER BY deptno ;
```

DEPTNO	CLERK	MANAGER	PRESIDENT	ANALYST	SALESMAN
10	1	1	1	0	0
20	2	1	0	2	0
30	1	1	0	0	4

```
SCOTT>
```


3. SQL 복수행 함수

③ EMP 테이블에서 부서별로 각 직급별 인원이 몇 명인 지 계산해서 출력하세요

- PIVOT 함수를 이용하는 방법

```

oracle@localhost:~
SCOTT>SELECT * FROM (SELECT deptno, job, empno
2      FROM emp )
3  PIVOT
4  (
5    COUNT(empno)
6    FOR job IN ('CLERK' as "CLERK" ,
7                'MANAGER' as "MANAGER" ,
8                'PRESIDENT' as "PRESIDENT" ,
9                'ANALYST' as "ANALYST" ,
10               'SALESMAN' as "SALESMAN")
11 )
12 ORDER BY deptno ;

```

DEPTNO	CLERK	MANAGER	PRESIDENT	ANALYST	SALESMAN
10	1	1	1	0	0
20	2	1	0	2	0
30	1	1	0	0	4

```

SCOTT>

```

3. SQL 복수행 함수

- PIVOT 부분에 조건을 여러 개 사용하기

```
oracle@localhost:~  
SCOTT>SELECT * FROM (SELECT deptno,job,empno,sal  
2          FROM emp)  
3 PIVOT  
4 (  
5   COUNT(empno) as CNT ,  
6   SUM(NVL(sal,0)) as S_SAL FOR job IN ('CLERK' as "C",  
7                                         'MANAGER' as "M",  
8                                         'PRESIDENT' as "P",  
9                                         'ANALYST' as "A",  
10                                        'SALESMAN' as "S")  
11 )  
12 ORDER BY deptno ;
```

DEPTNO	C_CNT	C_S_SAL	M_CNT	M_S_SAL	P_CNT	P_S_SAL	A_CNT	A_S_SAL	S_CNT	S_S_SAL
10	1	1300	1	2450	1	5000	0		0	
20	2	1900	1	2975	0		2	6000	0	
30	1	950	1	2850	0		0		4	5600

```
SCOTT>
```

3. SQL 복수행 함수

6) UNPIVOT 함수

```

oracle@localhost:~
SCOTT>CREATE TABLE t_unpivot AS
2      ( SELECT * FROM (SELECT deptno,job,empno
3                        FROM emp )
4        PIVOT
5        (
6          COUNT(empno)
7          FOR job IN ('CLERK' as "CLERK"
8                     , 'MANAGER' as "MANAGER"
9                     , 'PRESIDENT' as "PRESIDENT"
10                    , 'ANALYST' as "ANALYST"
11                    , 'SALESMAN' as "SALESMAN")
12        )
13      );
Table created.
SCOTT>SELECT * FROM t_unpivot ;

  DEPTNO      CLERK      MANAGER  PRESIDENT      ANALYST      SALESMAN
-----
      30          1          1          0          0          4
      20          2          1          0          2          0
      10          1          1          1          0          0
SCOTT>

```

PIVOT 테이블 생성

3. SQL 복수행 함수

```
oracle@localhost:~
SCOTT>set pagesize 50
SCOTT>SELECT *
2 FROM t_unpivot
3 UNPIVOT (
4 empno FOR job IN (CLERK,MANAGER,PRESIDENT,ANALYST,SALESMAN)
5 ) ;
```

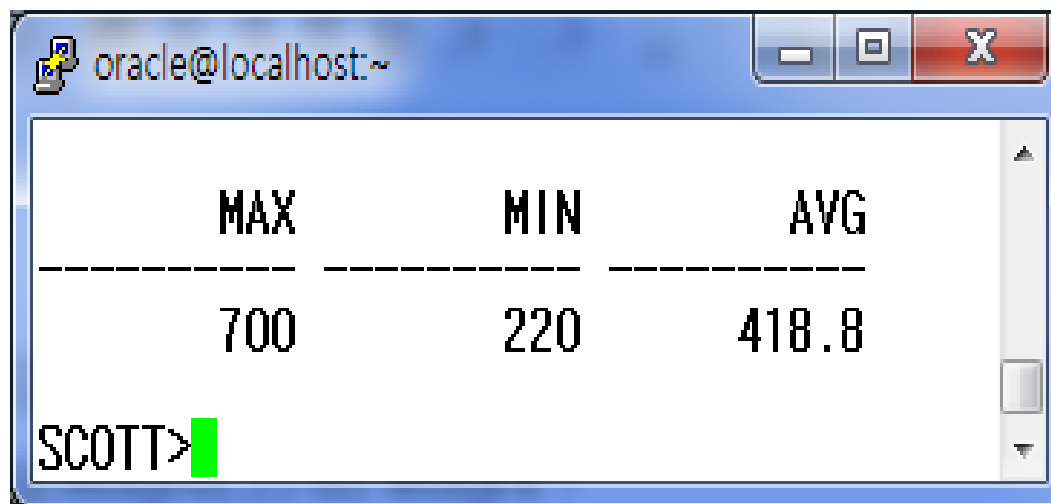
DEPTNO	JOB	EMPNO
30	CLERK	1
30	MANAGER	1
30	PRESIDENT	0
30	ANALYST	0
30	SALESMAN	4
20	CLERK	2
20	MANAGER	1
20	PRESIDENT	0
20	ANALYST	2
20	SALESMAN	0
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
10	ANALYST	0
10	SALESMAN	0

```
15 rows selected.
SCOTT>
```

3. SQL 복수행 함수

6. 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.



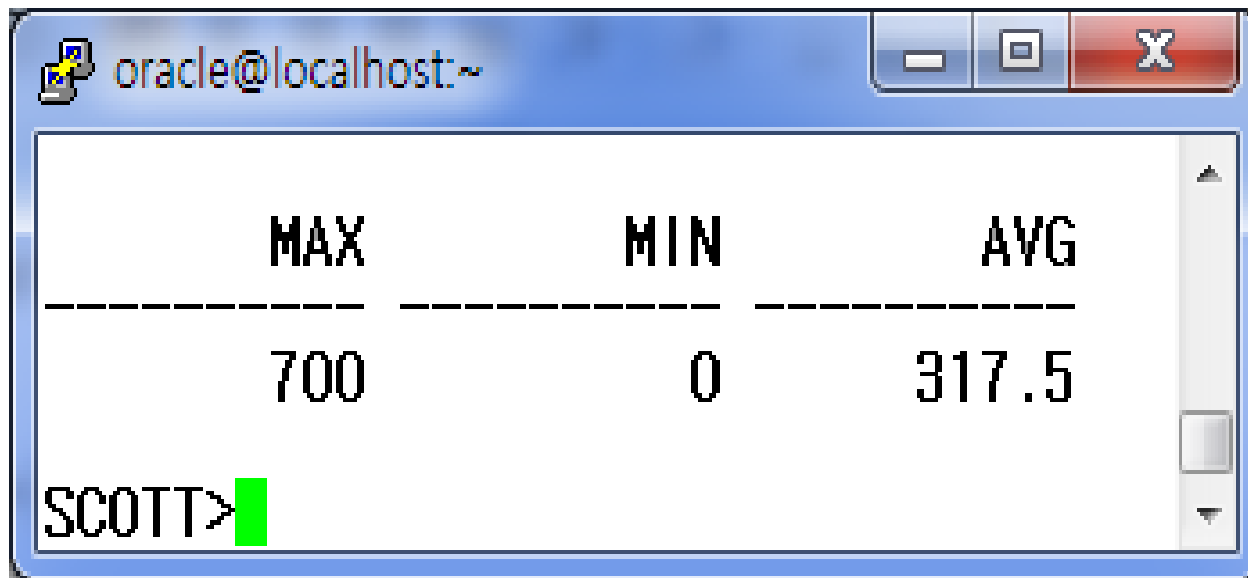
The screenshot shows a terminal window titled 'oracle@localhost:~'. It displays the output of a SQL query. The output is a table with three columns: MAX, MIN, and AVG. The values are 700, 220, and 418.8 respectively. The prompt 'SCOTT>' is visible at the bottom.

MAX	MIN	AVG
700	220	418.8

SCOTT>

3. SQL 복수행 함수

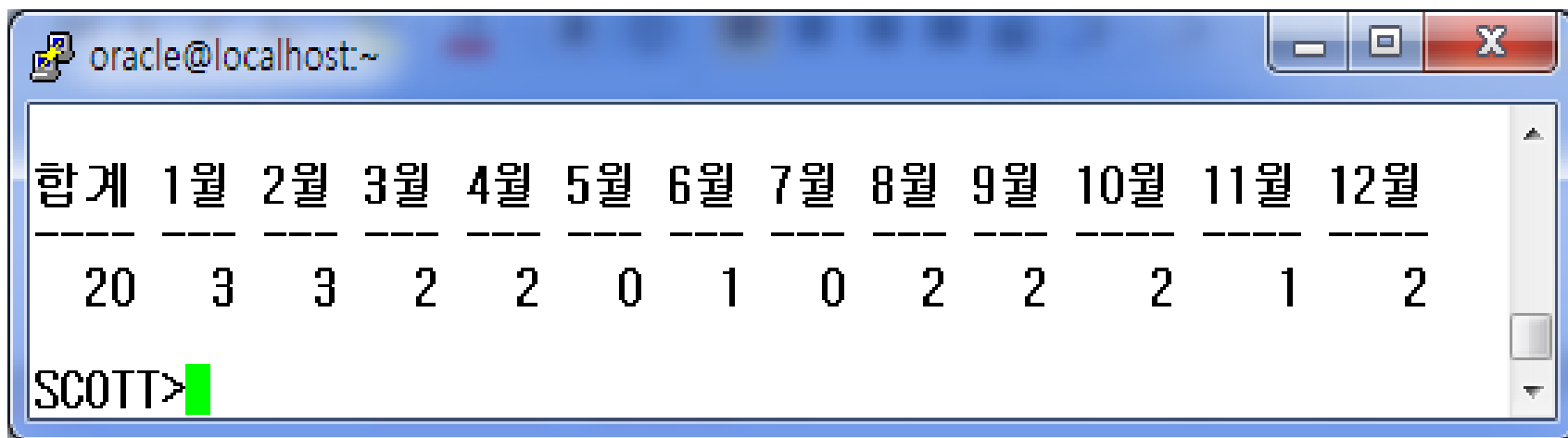
2) Professor 테이블을 사용하여 교수 중에서 급여(Pay)와 보너스(bonus)를 합친 금액이 가장 많은 경우와 가장 적은 경우, 평균 금액을 구하세요. 단 보너스가 없을 경우는 급여를 0 으로 계산하고 출력 금액은 모두 소수점 첫째 자리까지만 나오게 하세요.



```
oracle@localhost:~  
  
      MAX      MIN      AVG  
-----  
      700        0    317.5  
  
SCOTT>
```

3. SQL 복수행 함수

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



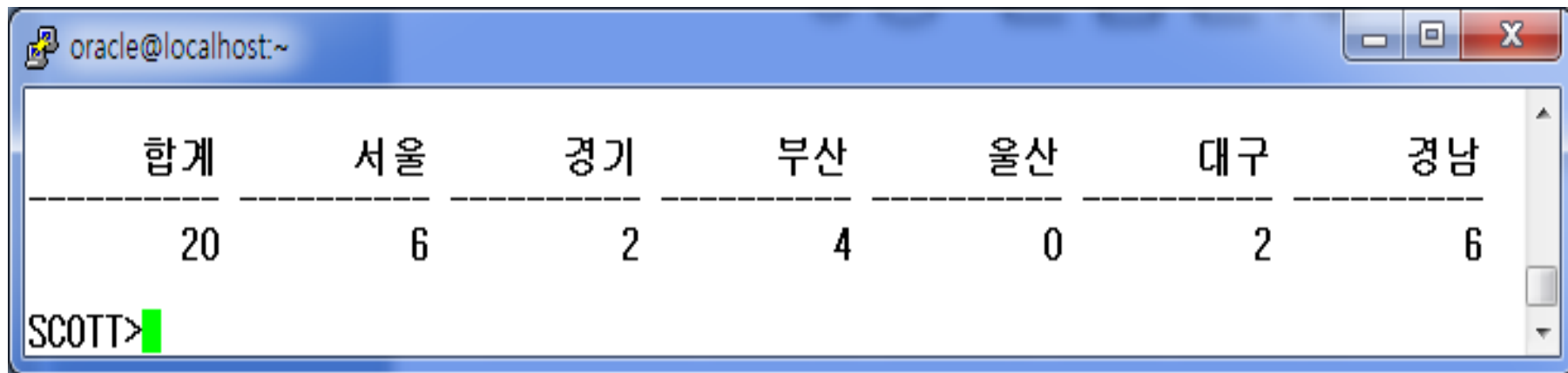
oracle@localhost:~

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>

3. SQL 복수행 함수

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



oracle@localhost:~

합계	서울	경기	부산	울산	대구	경남
20	6	2	4	0	2	6

SCOTT>

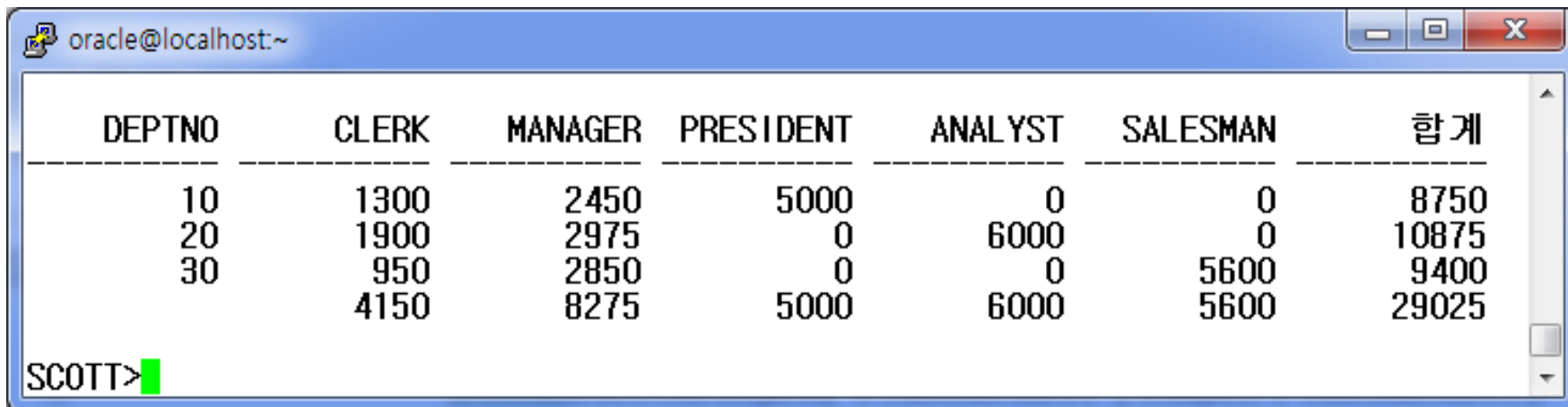
3. SQL 복수행 함수

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;



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

SCOTT>

3. SQL 복수행 함수

6) Professor 테이블의 교수번호와 이름을 아래의 예시 화면 형태로 출력하세요.

```

oracle@localhost:~
-----
      NO      사번1 이름1      사번2 이름2      사번3 이름3
-----
        1      1001 조인형      1002 박승곤      1003 송도권
        2      2001 양선희      2002 김영조      2003 주승재
        3      3001 김도형      3002 나한열      3003 김현정
        4      4001 심슨범      4002 최슬기      4003 박원범
        5      4004 차범철      4005 바비      4006 전민
        6      4007 허은      *****      *****
6 rows selected.
SCOTT>

```

3. SQL 복수행 함수

7. 그 외 주요 그룹 함수

3. SQL 복수행 함수

1) **LAG 함수** :이전 행 값을 가져 올 때 사용하는 함수입니다.

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

oracle@localhost:~

```
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
허은희	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>

oracle@localhost:~

```
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
허은희	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>

3. SQL 복수행 함수

2) LEAD 함수

LEAD 함수는 LAG 함수와 반대로 이후의 값을 가져오는 함수입니다.

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
심스	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
허은	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
심스	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
허은	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.

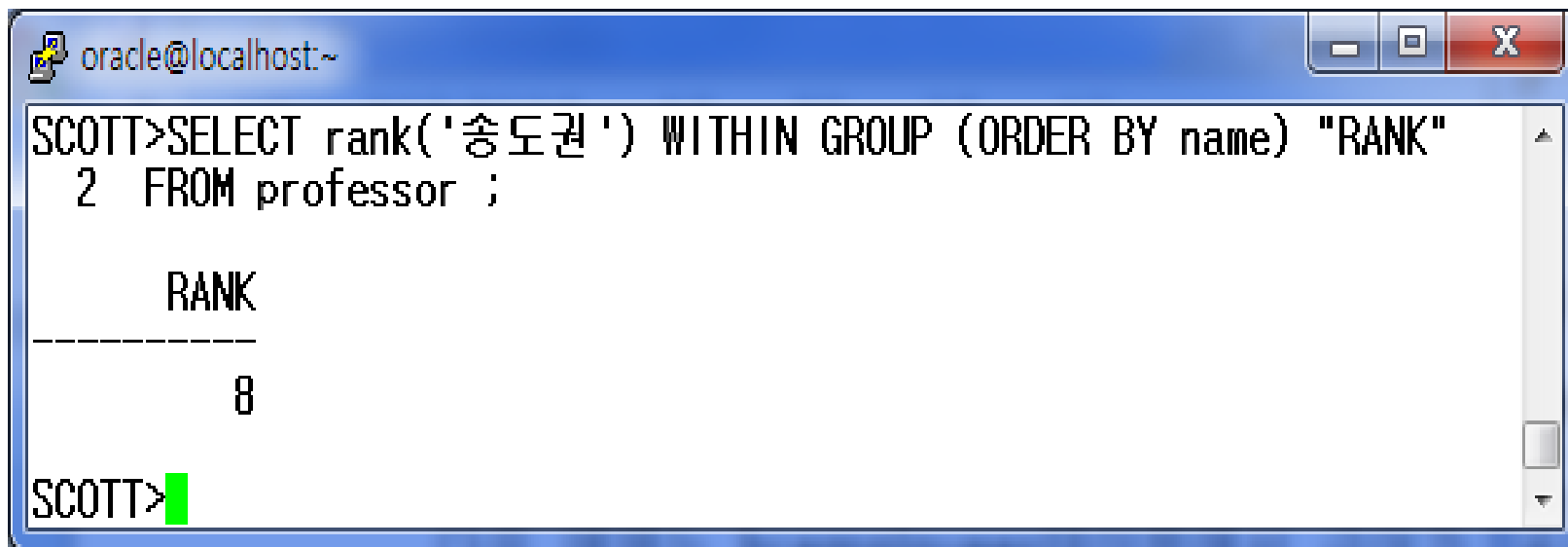
SCOTT>

3. SQL 복수행 함수

3) 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>
  
```

3. SQL 복수행 함수

3) RANK 함수 - 집계용

RANK () (ORDER BY 조건컬럼명 [ASC | DESC])

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

```
oracle@localhost:~
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 ;
```

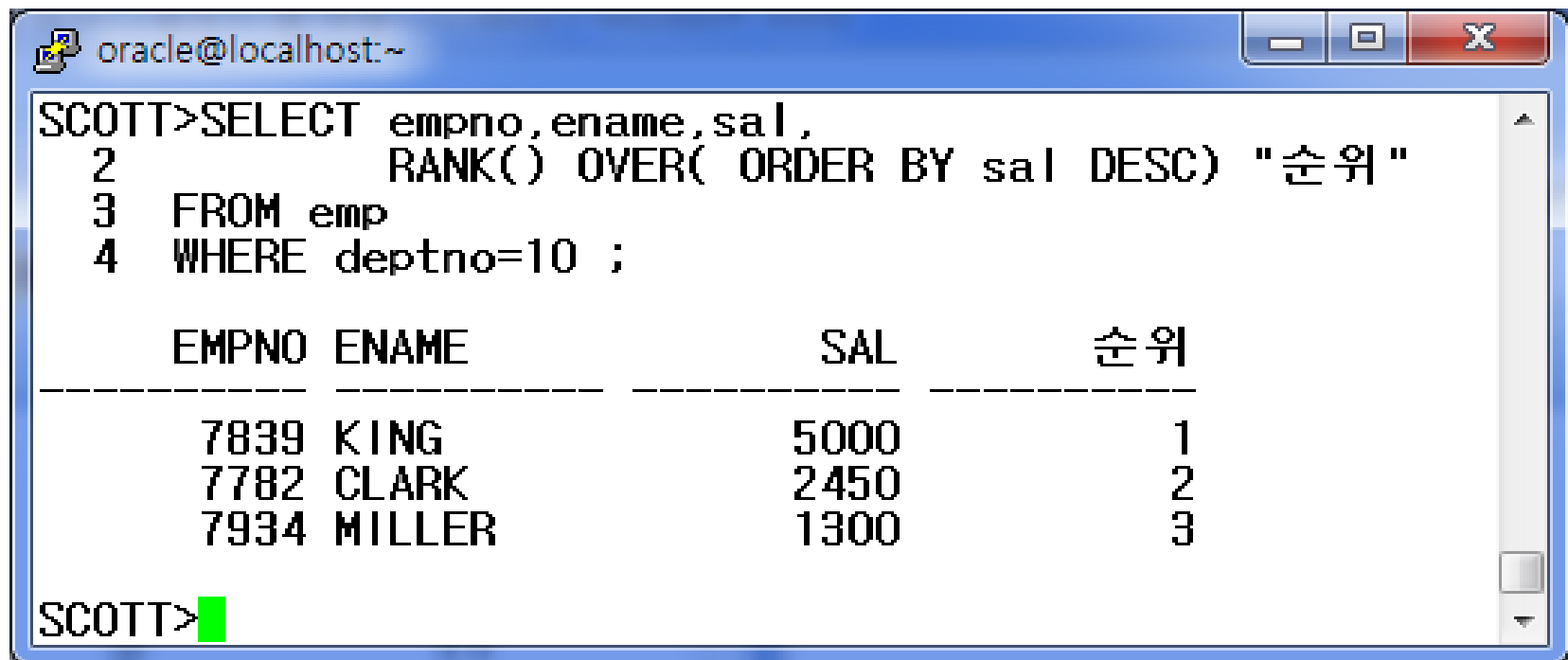
PROFNO	NAME	PAY	RANK	RANK_DESC
4001	심승민	570	16	1
1001	조인형	550	15	2
3001	김도형	530	14	3
4005	바비	500	13	4
2003	주승재	490	12	5
1002	박승조	380	11	6
2002	김영열	350	10	7
3002	나한열	330	8	8
4002	최슬기	330	8	8
4003	박원범	310	7	10
4007	허현정	290	5	11
3003	김현정	290	5	11
1003	송도권	270	4	13
4004	차범철	260	3	14
2001	양선희	250	2	15
4006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

16 rows selected.

```
SCOTT>
```

3. SQL 복수행 함수

② Emp 테이블에서 10번 부서에 속한 직원들의 사번과 이름, 급여, 해당 부서내의 급여순위를 출력하세요.



```
oracle@localhost:~  
SCOTT>SELECT empno,ename,sal,  
2          RANK() OVER( ORDER BY sal DESC) "순위"  
3 FROM emp  
4 WHERE deptno=10 ;
```

EMPNO	ENAME	SAL	순위
7839	KING	5000	1
7782	CLARK	2450	2
7934	MILLER	1300	3

```
SCOTT>
```


3. SQL 복수행 함수

③ emp 테이블을 사용하여 사번, 이름, 급여, 부서번호, 부서별 급여순위를 출력하세요.

```

oracle@localhost:~
SCOTT>SELECT empno,ename,sal,deptno,
2          RANK() OVER (PARTITION BY deptno
3                        ORDER BY sal desc ) "RANK"
4 FROM emp ;

```

EMPNO	ENAME	SAL	DEPTNO	RANK
7839	KING	5000	10	1
7782	CLARK	2450	10	2
7934	MILLER	1300	10	3
7902	FORD	5000	20	1
7788	SCOTT	3000	20	2
7566	JONES	2975	20	3
7876	ADAMS	1100	20	4
7369	SMITH	800	20	5
7698	BLAKE	2850	30	1
7499	ALLEN	1600	30	2
7844	TURNER	1500	30	3
7521	WARD	1250	30	4
7654	MARTIN	1250	30	4

```

13 rows selected.
SCOTT>

```

3. SQL 복수행 함수

- ④ emp 테이블을 사용하여 사번, 이름, 급여, 부서번호, 부서 내 job별로 급여순위를 출력하세요

```
oracle@localhost:~
SCOTT>SELECT empno,ename,sal,deptno,job,
2          RANK() OVER (PARTITION BY deptno,job
3                      ORDER BY sal desc ) "RANK"
4 FROM emp ;
```

EMPNO	ENAME	SAL	DEPTNO	JOB	RANK
7934	MILLER	1300	10	CLERK	1
7782	CLARK	2450	10	MANAGER	1
7839	KING	5000	10	PRESIDENT	1
7902	FORD	5000	20	ANALYST	1
7788	SCOTT	3000	20	ANALYST	2
7876	ADAMS	1100	20	CLERK	1
7369	SMITH	800	20	CLERK	2
7566	JONES	2975	20	MANAGER	1
7698	BLAKE	2850	30	MANAGER	1
7499	ALLEN	1600	30	SALESMAN	1
7844	TURNER	1500	30	SALESMAN	2
7521	WARD	1250	30	SALESMAN	3
7654	MARTIN	1250	30	SALESMAN	3

```
13 rows selected.
SCOTT>
```

3. SQL 복수행 함수

4)누적 합계 구하기

① panmae 테이블을 사용하여 1000 번 대리점의 판매 내역을 출력하되 판매일자, 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>
```

3. SQL 복수행 함수

② panmae 테이블을 사용하여 1000 번 대리점의 판매 내역을 판매일자별로 분류하고 같은 일자일 경우 제품 코드별로 한번 더 분류한 후 판매일자, 제품코드, 판매량, 판매금액, 누적판매금액을 아래와 같이 출력하세요.

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

판매일자	제품코드	판매량	판매금액	누적판매금액
20110103	100	2	1600	1600
20110101	100	3	2400	4000
20110102	102	2	2000	2000
20110102	105	2	3000	3000

```
SCOTT>
```