1. **아래는 부서별 급여합계와 전체 합계를 조회하는 예제이다.**

**ROLL UP을 이용하여 UNION ALL예제와 동일한 결과를 조회하는 쿼리를 작성하시오**

SELECT deptno, SUM(sal) s\_sal

FROM emp

GROUP BY deptno

UNION ALL

SELECT NULL deptno

, SUM(sal) s\_sal

FROM emp

;

DEPTNO S\_SAL

-------- -------------

30 9400

20 10875

10 8750

29025

* SELECT deptno

, SUM(sal) s\_sal

FROM emp

GROUP BY

;

1. **순위함수에 대한 설명 중 틀린 것은 무엇인가?**  
   1) RANK 함수는 순위를 부여하는 함수로 동일순위 처리가 가능하다.  
   2) DENSE\_RANK 함수는 RANK함수와 같은 역할을 하지만 동일 등수 순위에 영향이 없다.   
   3) ROW\_NUMBER 함수는 특정 순위로 일련번호를 제공하는 함수로 동일 순위 처리가 불가능하다.   
   4) 순위 함수 사용 시 ORDER BY 절은 입력하지 않아도 된다.
2. **다음의 결과가 나오도록 빈칸에 알맞은 함수를 쓰시오  
   [문제] 다음과 같이 월별 금액을 가진 집합에서 해당 월을 기준으로 이전 3개월간 금액 합계를 출력하는 SQL을 완성하세요.**

원본 리스트

WITH t AS

(

SELECT '201601' yyyymm, 100 amt FROM dual

UNION ALL SELECT '201602', 200 FROM dual

UNION ALL SELECT '201603', 300 FROM dual

UNION ALL SELECT '201604', 400 FROM dual

UNION ALL SELECT '201605', 500 FROM dual

)

SELECT \* FROM t

;

<원본테이블>

|  |  |
| --- | --- |
| YYYYMM | AMT |
| 201601 | 100 |
| 201602 | 200 |
| 201603 | 300 |
| 201604 | 400 |
| 201605 | 500 |

<결과테이블>

|  |  |  |
| --- | --- | --- |
| YYYYMM | AMT | AMT\_PRE3 |
| 201601 | 100 |  |
| 201602 | 200 | 100 |
| 201603 | 300 | 300 |
| 201604 | 400 | 600 |
| 201605 | 500 | 900 |

SELECT yyyymm

, amt

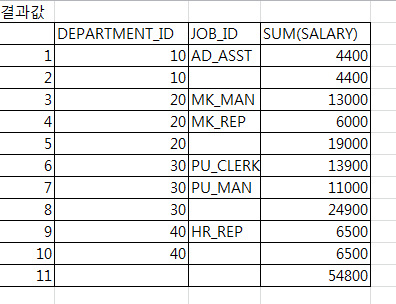
, \_\_\_\_\_\_\_\_\_\_(amt) OVER(ORDER BY TO\_DATE(yyyymm,'yyyymm')

RANGE BETWEEN INTERVAL '3' MONTH \_\_\_\_\_\_\_\_\_\_\_\_\_

AND INTERVAL '1' MONTH \_\_\_\_\_\_\_\_\_\_) amt\_pre3

FROM t;

1. **소계, 중계, 합계처럼 계층적 분류를 포함하고 있는 데이터의 집계에 적합한 GROUP 함수 두 가지는 무엇인가?**
2. ROLLUP, SUM
3. ROLLUP, CUBE
4. GROUPING, SUM
5. CUBE, SUM
6. **그룹 내 순위관련 WINDOW 함수의 특징이 틀린 것은 무엇인가?**
7. RANK함수는 동일한 값에 대해서는 동일한 순위를 부여한다.  
    (같은 등수에 여럿이 존재하는 경우 등수가 SKIP될 수 있음)
8. DENSE\_RANK 함수는 RANK 함수와 흡사하나, 동일한 순위를 하나의 건수로 취급하는 것이 틀린 점이다. (같은 등수에 여럿이 존재하는 경우에도 등수가 SKIP 되지 않음)
9. CUMM\_RANK 함수는 누적된 순위를 부여할 수 있다. (등수를 누적 순위로 표현함)
10. RANK 함수가 동일한 값에 대해서는 동일한 순위를 부여하는데 반해, ROW\_NUMBER 함수는 고유한 순위를 부여한다. (같은 등수가 존재할 수 없음)
11. **다음 Query 결과값을 만족하기 위해서 빈칸의 적합한 함수를 작성하시오.**



<쿼리>

SELECT department\_id, job\_id, sum(salary)

from employees

where department\_id <= 40

group by ( )

order by department\_id;

1. **부서-Job-매니저 별 집계와, 부서-Job 별 집계와, Job-매니저 별 집계를 Grouping sets 함수를 이용해서 구하려 한다. 1,2,3번에 들어갈 알맞은 값은?**

|  |
| --- |
| **SELECT d.dname,**  **e.job,**  **e.mgr,**  **SUM(e.sal) "Total Sal"**  **FROM emp e,**  **dept d**  **WHERE 1=1**  **AND e.deptno = d.deptno**  **GROUP BY GROUPING SETS**  **(**  **( 1 ). ( 2 ), ( 3 )**  **)**  **;** |

1. **( )**
2. **( )**
3. **( )**
4. **다음은 윈도우 함수에 설명이다. 현재행을 기준으로 파티션 내에서 앞의 한건, 현재 행, 뒤의 한건을 범위를 지정하는 Over 의 옵션은?**

1. ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
2. RANGE UNBOUNDED PRECEDING
3. ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING
4. ROWS BETWEEN 1 AND 2
5. **Grouping 의 경우, Rollup 이나 Cube 에 의한 소계가 계산된 결과에는 ( ) 이 표시되고 그 외에는 ( ) 으로 표기된다.**
6. **아래와 같은 결과값이 나오도록 빈 칸을 채우시오.**

SELECT deptno

, empno

, SUM(sal) s\_sal

FROM emp

GROUP BY **( )**

DEPTNO EMPNO S\_SAL

--------------------------------------------

10 7782 2450

10 7839 5000

10 7934 1300

10 8750

20 7369 800

20 7566 2975

20 7788 3000

20 7876 1100

20 7902 3000

20 10875

30 7900 950

30 7499 1600

30 7521 1250

30 7654 1250

30 7698 2850

30 7844 1500

30 9400

29025

1. **아래 결과가 나오는 알맞은 순위 함수를 선택하시오.**

**<보기>**

1. **DENSE\_RANK**
2. **ROW\_NUMBER**
3. **RANK**

SELECT deptno, empno, sal

, **( )** ( ) OVER(ORDER BY sal DESC) A

, **( )** ( ) OVER(ORDER BY sal DESC) B

, **( )** ( ) OVER(ORDER BY sal DESC) C

FROM emp

;

DEPTNO EMPNO SAL A B C

------- ---------- ---------- ---------- ---------- ---------- -----------------

10 7839 5000 1 1 1

20 7788 3000 2 2 2

20 7902 3000 2 2 3

20 7566 2975 4 3 4

30 7698 2850 5 4 5

10 7782 2450 6 5 6

30 7499 1600 7 6 7

30 7844 1500 8 7 8

10 7934 1300 9 8 9

1. **아래 결과가 나오는 알맞은 값을 서술하시오.**

SELECT empno, ename, deptno,

SUM(sal) OVER ( ) A

SUM(sal) OVER **( )** B

SUM(sal) OVER **( )** C

FROM emp;

EMPNO ENAME DEPTNO A B C

------ ------- ---------- ---------- ---------- ---------- --------------------

7782 CLARK 10 29025 8750 2450

7839 KING 10 29025 8750 7450

7934 MILLER 10 29025 8750 8750

7369 SMITH 20 29025 10875 800

7566 JONES 20 29025 10875 3775

7788 SCOTT 20 29025 10875 6775

7876 ADAMS 20 29025 10875 7875

7902 FORD 20 29025 10875 10875

7499 ALLEN 30 29025 9400 1600

7521 WARD 30 29025 9400 2850

7654 MARTIN 30 29025 9400 4100

7698 BLAKE 30 29025 9400 6950

7844 TURNER 30 29025 9400 8450

7900 JAMES 30 29025 9400 9400

1. **아래 쿼리에서 빈 칸을 채우세요. (Alias 참조)**

<Query>

SELECT mp.organization\_code

,msi.segment1

,epi.weight "무게"

,RANK() OVER (( )epi.weight( )) AS "가벼운 순서"

,RANK() OVER (( )epi.organization\_id ORDER BY

epi.weight) AS "사업장별 무거운 순서”

,( ) OVER (ORDER BY epi.weight) AS "동순위 인정없는 무게순"

FROM ebom\_packing\_items epi

,mtl\_system\_items\_b msi

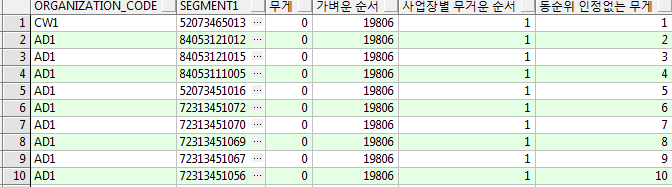
,mtl\_parameters mp

WHERE 1=1

AND epi.organization\_id = msi.organization\_id

AND epi.inventory\_item\_id = msi.inventory\_item\_id

AND msi.organization\_id = mp.organization\_id



1. **다음 쿼리는 어떠한 값을 구하기 위한 쿼리인가?**

**(단, papf.start\_date를 입사일이라고 본다.)**  
SELECT papf.full\_name

,papf.start\_date

,LAG(papf.full\_name,1) OVER (ORDER BY papf.start\_date)

,LAG(papf.start\_date,1) OVER (ORDER BY papf.start\_date)

FROM per\_all\_people\_f papf

,fnd\_user fu

WHERE 1=1

AND papf.person\_id = fu.employee\_id

ORDER BY papf.start\_date

1) 입사일이 본인보다 한 명 늦은 사람에 대한 이름과 입사일을 보여준다.  
2) 입사일이 빠른 순서대로 정렬하며, 입사일이 본인과 동일한 사람에 대한 이름과 입사일을 보여준다.

3) 입사일이 늦은 순서대로 정렬하며, 입사일이 본인보다 한 명 앞선 사람에 대한 이름과 입사일을 보여준다.

4) 입사일이 빠른 순서대로 정렬하며, 입사일이 본인보다 한 명 앞선 사람에 대한 이름과 입사일을 보여준다.

1. **아래 쿼리는 무엇을 구하기 위한 쿼리인 지 간단하게 설명하세요.**

SELECT bqh.dept\_code AS "부서코드"

,bqh.dept\_name AS "부서명"

,SUM(bqh.quoted\_amt) AS "견적금액"

FROM bem\_quote\_header\_v bqh

WHERE 1=1

AND bqh.status\_code = 'ORDERED'

AND bqh.quoted\_date >= TRUNC(SYSDATE,'y')

GROUP BY ROLLUP(bqh.dept\_code,bqh.dept\_name)

HAVING SUM(bqh.quoted\_amt) >= 10000000

ORDER BY bqh.dept\_code,bqh.dept\_name

[

]

1. **소계 /중계/합계처럼 계층적 분류를 포함하고 있는 데이터의 집계에 적합한 GROUP 함 수 두가지는?**

( , )

1. **MIN 함수와 동일하게 사용할 수 있는 WINDOW 함수는?**
2. LAG
3. RATIO\_TO\_REPORT
4. LEAD
5. FIRST\_VALUE
6. CUME\_DIST
7. **아래 SQL 의 결과 값은?**

|  |
| --- |
| 테이블 A  NAME SAL  -----------------------------  SCG 1000  KDH 3000 |
| SELECT A.NAME , A.SAL , ROUND(RATIO\_TO\_REPORT(SAL) OVER(),2) AS R  FROM A  WHERE A.NAME = ‘SCG; |

( )

1. **상자 안의 SQL과 같은 결과를 도출하는 SQL문은?**

|  |
| --- |
| SELECT first\_value(msi.creation\_date)  OVER (PARTITION BY msi.inventory\_item\_id  ORDER BY msi.creation\_date)  FROM mtl\_system\_items\_b msi  WHERE 1=1; |

1. SELECT MAX(msi.creation\_date)

FROM mtl\_system\_items\_b msi;

1. SELECT MIN(msi.creation\_date)

FROM mtl\_system\_items\_b msi;

1. SELECT COUNT(msi.creation\_date)

FROM mtl\_system\_items\_b msi;

1. SELECT AVG(msi.inventory\_item\_id)

FROM mtl\_system\_items\_b msi;

1. SELECT SUM(msi.inventory\_item\_id)

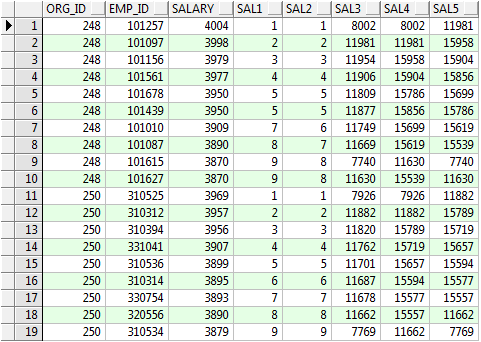
FROM mtl\_system\_items\_b msi;

1. **그룹 내 행 순서 관련 함수에 속하지 않는 함수를 모두 고르시오.**
2. FIRST\_VALUE
3. LAST\_VALUE
4. RANK
5. DENSE\_RANK
6. LAG
7. **아래 쿼리를 실행하였을 때의 결과를 빈칸에 입력하시오.**

|  |
| --- |
| SELECT ood.ORGANIZATION\_CODE  ,NTILE(4) OVER (ORDER BY ood.organization\_id) AS org\_ntile  FROM org\_organization\_definitions ood; |

|  |  |
| --- | --- |
| ORGANIZATION\_CODE | ORG\_NTILE |
| V01 |  |
| M00 |  |
| M01 |  |
| M02 |  |
| M03 |  |
| M04 |  |
| M05 |  |
| R01 |  |
| R02 |  |
| R03 |  |
| M06 |  |
| M07 |  |
| M08 |  |
| R04 |  |
| R05 |  |
| E01 |  |
| CD1 |  |
| CW1 |  |
| CW9 |  |
| CH1 |  |
| AD1 |  |
| NA1 |  |
| JT1 |  |
| UC1 |  |
| CD2 |  |

1. 다음 결과값이 출력되도록 빈칸을 채우시오.



<Query>

select x.org\_id,

x.employee\_id as emp\_id,

x.total\_basic\_salary\_amount as salary,

( ) as sal1,

( ) as sal2,

( ) as sal3,

( ) as sal4,

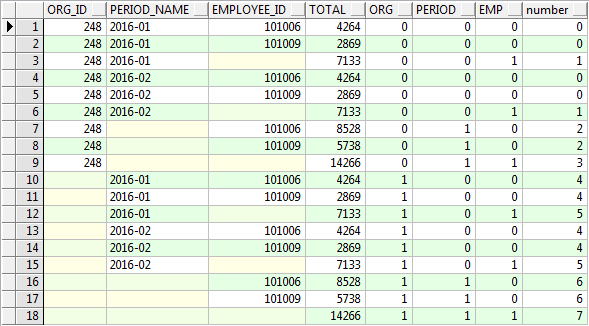
( ) as sal5

from ehm\_salary\_calculation x

where x.period\_name = '2016-02'

and x.total\_basic\_salary\_amount between 3870 and 4004

1. 다음 결과값이 출력되도록 빈칸을 채우시오.



<Query>

select x.org\_id,

x.period\_name,

x.employee\_id,

sum(x.sal01) as total,

( ) as org,

( ) as period,

( ) as emp,

( ) as "number"

from ehm\_salary\_calculation x

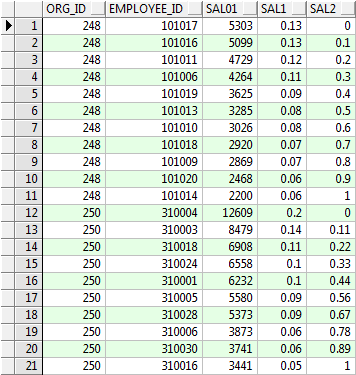
where x.period\_name between '2016-01' and '2016-03'

and x.employee\_id in (101006, 101009)

group by ( )

order by x.org\_id, x.period\_name, x.employee\_id

1. 다음 결과값이 출력되도록 빈칸을 채우시오



<Query>  
select x.org\_id,

x.employee\_id,

x.sal01,

( ) as sal1,

( ) as sal2

from ehm\_salary\_calculation x

where x.period\_name = '2016-02'

--and x.org\_id = 250

and ((x.employee\_id between 310001 and 310030) or (x.employee\_id between 100001 and 101020))

<정답>

1. rollup(deptno)

2. 4

3. sum, precending, following

4. 2

5. 3

6. rollup(department\_id, job\_id)

7. 1) (dname, job, mgr)  
 2) (dname, job)  
 3) (job, mgr)

8. 1

9. 1, 0

10. rollup(deptno, empno)

11. ㄷ, ㄱ, ㄴ

12. partition by deptno, partition by empmo

13. ORDER BY, DESC, PARTITION BY, ROW\_NUMBER()

14. 4

15. 금년도 생성된 견적 중에서 주문으로 생성되었고, 부서별 견적 총 금액의 합계가 1천만이 넘는 건들에 대하여 부서별 견적 총 금액을 보고자 함. (WHERE 절에 대한 설명은 없어도 ROLL UP이랑 HAVING에 대한 설명만 있으면 정답)

16. CUBE/SUM

17. 4

18. SCG 1000 1

19. 3

20. 3, 4

21. 1 \* 7개, 2 \* 7개, 3 \* 7개, 4 \* 4개

22. select x.org\_id,

x.employee\_id as emp\_id,

x.total\_basic\_salary\_amount as salary,

rank() over(partition by x.org\_id order by x.total\_basic\_salary\_amount desc) as sal1,

dense\_rank() over(partition by x.org\_id order by x.total\_basic\_salary\_amount desc) as sal2,

round(sum(x.total\_basic\_salary\_amount) over (partition by org\_id order by x.total\_basic\_salary\_amount rows between 1 preceding and 1 following)) as sal3,

round(sum(x.total\_basic\_salary\_amount) over (partition by org\_id order by x.total\_basic\_salary\_amount rows between 1 preceding and 2 following)) as sal4,

round(sum(x.total\_basic\_salary\_amount) over (partition by org\_id order by x.total\_basic\_salary\_amount rows between 2 preceding and 1 following)) as sal5

from ehm\_salary\_calculation x

where x.period\_name = '2016-02'

and x.total\_basic\_salary\_amount between 3870 and 4004

23. select x.org\_id,

x.period\_name,

x.employee\_id,

sum(x.sal01) as total,

grouping(x.org\_id) as org,

grouping(x.period\_name) as period,

grouping(x.employee\_id) as emp,

grouping\_id(x.org\_id, x.employee\_id) as "number1" ,

grouping\_id(x.org\_id, x.period\_name, x.employee\_id) as "number2"

from ehm\_salary\_calculation x

where x.period\_name between '2016-01' and '2016-03'

and x.employee\_id in (101006, 101009)

group by cube(x.org\_id, x.period\_name, x.employee\_id)

order by x.org\_id, x.period\_name, x.employee\_id

24. select x.org\_id,

x.employee\_id,

x.sal01,

round(ratio\_to\_report(x.sal01) over (partition by x.org\_id), 2) as sal1,

round(percent\_rank() over (partition by x.org\_id order by x.sal01 desc), 2) as sal2

from ehm\_salary\_calculation x

where x.period\_name = '2016-02'

--and x.org\_id = 250

and ((x.employee\_id between 310001 and 310030) or (x.employee\_id between 100001 and 101020))