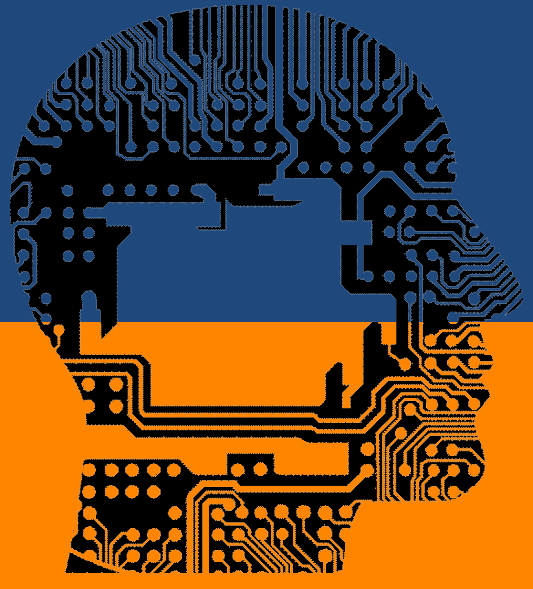
한국IT진흥부설 정보보호교육학원 아이섹

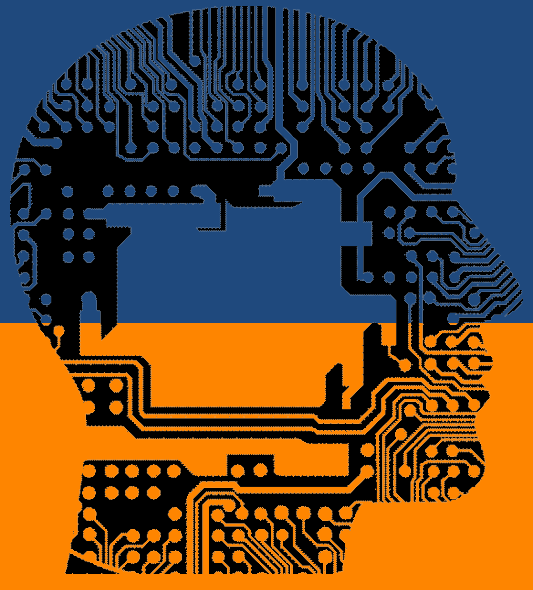


JAVA 웹 개발자 양성과정
DataBase - SQLD

3강 - 서브쿼리, 그룹함수, 윈도우함수

By SoonGu Hong

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정

DataBase

1. 서브 쿼리

➢ 서브 쿼리란

① 서브 쿼리(Subquery)란 하나의 SQL문안에 포함되어 있는 또 다른 SQL문을 말한다.
② 조인은 조인에 참여하는 모든 테이블이 대등한 관계에 있기 때문에 조인에 참여하는 모든 테이블의 칼럼을 어느 위치에서라도 자유롭게 사용할 수 있다. 그러나 서브 쿼리는 메인 쿼리의 칼럼을 모두 사용할 수 있지만 메인 쿼리는 서브 쿼리의 칼럼을 사용할 수 없다.

➢ 서브 쿼리 사용시 주의점

① 서브쿼리를 괄호로 감싸서 사용한다.
② 서브 쿼리는 단일 행(Single Row) 또는 복수 행(Multiple Row) 비교 연산자와 함께 사용 가능하다.
③ 단일 행 비교 연산자는 서브 쿼리의 결과가 반드시 1건 이하이어야 하고 복수 행 비교 연산자는 서브 쿼리의 결과 건수와 상관 없다.
④ 서브쿼리에서는 ORDER BY를 사용하지 못한다. ORDER BY절은 SELECT절에서 오직 한 개만 올 수 있기 때문에 ORDER BY절은 메인 쿼리의 마지막 문장에 위치해야 한다.

➢ 서브 쿼리가 사용 가능한 위치

① SELECT 절 - FROM 절 - WHERE 절 - HAVING 절 - ORDER BY 절
② INSERT문의 VALUES 절 - UPDATE문의 SET 절

➢ 동작 방식에 따른 서브 쿼리 분류

동작 방식	설명
비 연관 서브 쿼리	• 서브 쿼리가 메인 쿼리의 칼럼을 가지고 있지 않은 형태의 서브 쿼리임 • 메인 쿼리에 값을 제공하기 위한 목적으로 주로 사용
연관 서브 쿼리	• 서브 쿼리가 메인 쿼리의 값을 가지고 있는 형태의 서브쿼리이다. • 일반적으로 메인 쿼리가 먼저 수행되어 읽혀진 데이터를 서브쿼리에서 조건이 맞는지 확인하고자 할 때 주로 사용한다.

➢ 반환 형태에 따른 서브 쿼리 분류

반환 형태	설명
단일 행 서브 쿼리	• 서브 쿼리의 실행 결과가 항상 1건 이하인 서브쿼리를 의미한다. • 항상 비교 연산자와 함께 사용된다. • (=, <, <=, >, >=, <>)
다중 행 서브 쿼리	• 서브 쿼리의 실행 결과가 여러 건인 서브쿼리를 의미한다. • 다중 행 서브 쿼리는 다중 행 비교 연산자와 함께 사용된다. • (IN, ALL, ANY, SOME, EXISTS)
다중 칼럼 서브 쿼리	• 서브 쿼리의 실행 결과로 여러 칼럼을 반환한다. • 메인 쿼리의 조건 절에 여러 칼럼을 동시에 비교 할 수 있다. • 서브 쿼리와 메인 쿼리의 칼럼 수와 칼럼 순서가 동일해야 한다

➢ 단일 행 서브 쿼리 실습

```
SELECT A.EMP_NO, A.EMP_NM, A.DEPT_CD
FROM TB_EMP A
WHERE A.DEPT_CD =
(
    SELECT DEPT_CD
    FROM TB_EMP
    WHERE EMP_NO = '1000000005'
);
```

❖ EMP_NO가 "1000000005"가 속한 팀의 팀원을 조회하는 SQL문

EMP_NO	EMP_NM	DEPT_CD
1000000002	이현승	100002
1000000003	이정수	100002
1000000004	이승준	100002
1000000005	김현희	100002

```
SELECT A.EMP_NO, B.EMP_NM, A.PAY_DE, A.PAY_AMT
FROM SQLD.TB_SAL_HIS A, TB_EMP B
WHERE A.PAY_DE = '20200525'
AND A.PAY_AMT >=
(
SELECT AVG(K.PAY_AMT)
  FROM SQLD.TB_SAL_HIS K
 WHERE K.PAY_DE = '20200525'
 )
 AND A.EMP_NO = B.EMP_NO;
```

❖ 2020년5월 기준 전체평균 이상의급여를 받고 있는 직원들의 리스트

EMP_NO	EMP_NM	PAY_DE	PAY_AMT
1000000003	이정수	20200525	4440000
1000000004	이승준	20200525	4910000
1000000005	김현희	20200525	4320000
1000000007	이순자	20200525	5000000
1000000009	박태범	20200525	4100000
1000000013	이나라	20200525	4340000
1000000015	이정직	20200525	5100000
1000000016	이진실	20200525	4340000
중간생략			
1000000031	김사랑	20200525	5150000
1000000032	이준표	20200525	5570000
1000000034	김익정	20200525	5890000
1000000035	최창수	20200525	4350000
1000000037	이현정	20200525	5140000
1000000040	김여진	20200525	4210000
9999999999	김회장	20200525	5470000

➢ 다중 행 서브 쿼리 실습

```
SELECT A.EMP_NO, COUNT(*) CNT
FROM TB_EMP_CERTI A
WHERE A.CERTI_CD
IN
(
SELECT K.CERTI_CD
FROM SQLD.TB_CERTI K
WHERE K.ISSUE_INSTI_NM = '한국데이터베이스진흥원'
)
GROUP BY A.EMP_NO
ORDER BY A.EMP_NO
;
```

❖ 한국데이터베이스진흥원에서 발급한 자격증을 가지고 있는 사원 번호 및 보유 자격 개수를 출력하는 SQL문

EMP_NO	CNT
1000000001	1
1000000002	1
1000000004	1
1000000005	2
1000000006	1
1000000007	1
1000000008	1
1000000009	2
중간생략...	
1000000034	2
1000000036	1
1000000037	1
1000000038	1
1000000040	1
9999999999	1

```
SELECT A.EMP_NO, COUNT(*) CNT
FROM TB_EMP_CERTI A
WHERE A.CERTI_CD
=
(
SELECT K.CERTI_CD
FROM SQLD.TB_CERTI K
WHERE K.ISSUE_INSTI_NM = '한국데이터베이스진흥원'
)
GROUP BY A.EMP_NO
ORDER BY A.EMP_NO
;
```

❖ 다중 행 서브 쿼리에 = 연산자를 써서 SQL에러가 발생함

```
SQL Error [1427] [21000]: ORA-01427: 단일 행 하위 질의에 2개 이상의 행이 리턴 되었습니다.
```

➢ 다중칼럼 서브쿼리 실습

```
SELECT A.EMP_NO
     , A.EMP_NM
     , A.DEPT_CD
     , B.DEPT_NM
     , A.BIRTH_DE
  FROM TB_EMP A
     , TB_DEPT B
WHERE (A.DEPT_CD, A.BIRTH_DE) IN
(
    SELECT K.DEPT_CD, MIN(K.BIRTH_DE) AS MIN_BIRTH_DE
    FROM TB_EMP K
    GROUP BY K.DEPT_CD
    HAVING COUNT(*) > 1
)
AND A.DEPT_CD = B.DEPT_CD
ORDER BY A.EMP_NO
;
```

❖ 한 부서에 2명 이상 있는 부서 중에서 각 부서의 생일 기준 최고참을 출력하는 SQL문

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM	BIRTH_DE
1000000004	이승준	100002	지원팀	19870623
1000000006	김혜진	100003	기획팀	19840715
1000000013	이나라	100004	디자인팀	19871201
1000000018	박바른	100006	데이터팀	19820629
1000000022	김순수	100007	개발팀	19871201
1000000026	김길정	100009	운영팀	19690524
1000000031	김사랑	100010	개발팀	19811201
1000000035	최창수	100012	인공지능팀	19690524
1000000040	김여진	100013	빅데이터팀	19871205

➢ EXISTS문 서브 쿼리 실습

```
SELECT A.DEPT_CD, A.DEPT_NM
FROM TB_DEPT A
WHERE EXISTS( SELECT 1 FROM TB_EMP K WHERE K.DEPT_CD = A.DEPT_CD
AND K.ADDR LIKE '%강남%')
;
```

❖ 직원들 중 주소가 강남인 직원이 소속된 부서 코드와 부서명을 출력

DEPT_CD	DEPT_NM
100009	운영팀
100010	개발팀

➢ 스칼라 서브 쿼리 실습

```
SELECT A.EMP_NO
     , (SELECT L.EMP_NM FROM TB_EMP L WHERE L.EMP_NO = A.EMP_NO) AS EMP_NM
     , A.CERTI_CD
     , (SELECT L.CERTI_NM FROM TB_CERTI L WHERE L.CERTI_CD = A.CERTI_CD) AS CERTI_NM
  FROM TB_EMP_CERTI A
 WHERE A.CERTI_CD
IN
(
SELECT K.CERTI_CD
FROM SQLD.TB_CERTI K
WHERE K.ISSUE_INSTI_NM = '한국데이터베이스진흥원'
)
ORDER BY CERTI_NM
;
```

❖ 한국데이터베이스진흥원에서 발급한 자격증을 가지고 있는 사원의 사원 번호, 사원명, 자격증 코드, 자격증명을 출력

EMP_NO	EMP_NM	CERTI_CD	CERTI_NM
1000000017	이겸손	100006	ADP
1000000033	홍사기	100006	ADP
1000000022	김순수	100006	ADP
1000000022	김순수	100006	ADP
1000000018	박바른	100006	ADP
1000000020	최정진	100006	ADP
1000000016	이진실	100006	ADP
중간생략...			
1000000027	박이수	100006	ADP
1000000013	이나라	100001	SQLD
1000000025	박호진	100001	SQLD
1000000005	김현희	100002	SQLP
1000000001	이경오	100002	SQLP
1000000024	박선영	100002	SQLP
1000000034	김익정	100002	SQLP
1000000008	김려원	100002	SQLP

➢ 인라인 뷰 서브 쿼리 실습

```
SELECT B.EMP_NO
, (SELECT L.EMP_NM FROM TB_EMP L WHERE L.EMP_NO = B.EMP_NO) AS EMP_NM
, B.CERTI_CD
, (SELECT L.CERTI_NM FROM TB_CERTI L WHERE L.CERTI_CD = B.CERTI_CD) AS CERTI_NM
FROM
(
SELECT K.CERTI_CD
FROM SQLD.TB_CERTI K
WHERE K.ISSUE_INSTI_NM = '한국데이터베이스진흥원'
) A
, TB_EMP_CERTI B
WHERE A.CERTI_CD = B.CERTI_CD
ORDER BY CERTI_NM
;
```

❖ 한국데이터베이스진흥원에서 발급한 자격증을 가지고 있는 사원의 사원 번호, 사원명, 자격증 코드, 자격증 명을 출력

EMP_NO	EMP_NM	CERTI_CD	CERTI_NM
1000000017	이겸손	100006	ADP
1000000033	홍사기	100006	ADP
1000000022	김순수	100006	ADP
1000000022	김순수	100006	ADP
1000000018	박바른	100006	ADP
1000000020	최정진	100006	ADP
1000000016	이진실	100006	ADP
1000000027	박이수	100006	ADP
1000000006	김혜진	100005	ADSP
1000000040	김여진	100005	ADSP
1000000032	이준표	100005	ADSP
1000000032	이준표	100005	ADSP
1000000004	이승준	100005	ADSP
1000000028	김나라	100005	ADSP
1000000019	박정혜	100005	ADSP
중간생략			
1000000007	이순자	100003	DASP
1000000031	김사랑	100003	DASP
1000000013	이나라	100001	SQLD
1000000025	박호진	100001	SQLD
1000000005	김현희	100002	SQLP
1000000001	이경오	100002	SQLP
1000000024	박선영	100002	SQLP
1000000034	김익정	100002	SQLP
1000000008	김려원	100002	SQLP

➢ HAVING절에서 서브 쿼리

```
SELECT B.DEPT_CD
     , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = B.DEPT_CD) "부서명"
     , AVG(A.PAY_AMT) "평균급여"
 FROM SQLD.TB_SAL_HIS A, TB_EMP B
WHERE A.PAY_DE = '20200525'
  AND A.EMP_NO = B.EMP_NO
GROUP BY B.DEPT_CD
HAVING AVG(A.PAY_AMT) >
(
    SELECT AVG(K.PAY_AMT)
      FROM SQLD.TB_SAL_HIS K, TB_EMP J
     WHERE K.PAY_DE = '20200525'
     AND K.EMP_NO = J.EMP_NO
     AND J.DEPT_CD = '100004'
 )
ORDER BY "평균급여" DESC
;
```

DEPT_CD	부서명	평균급여
100011	신사업본부	5570000
999999	회장실	5470000
100010	개발팀	4890000
100002	지원팀	4357500
100009	운영팀	4287500
100012	인공지능팀	4137500
100006	데이터팀	4105000
100013	빅데이터팀	3982500
100003	기획팀	3802500
100008	솔루션사업본부	3710000
100001	운영본부	3700000

❖ 2020년 05월의 "100004"(디자인팀) 부서의 평균 급여보다 평균급여가 높은 부서의 부서 코드와 부서명을 구하는 SQL문

➢ UPDATE문에 사용되는 서브 쿼리

```
ALTER TABLE TB_EMP ADD(DEPT_NM VARCHAR2(150));

UPDATE TB_EMP A
SET A.DEPT_NM = ( SELECT K.DEPT_NM FROM TB_DEPT K WHERE K.DEPT_CD = A.DEPT_CD)
;

COMMIT;
```

❖ TB_EMP테이블에 DEPT_NM 컬럼을 추가하고 UPDATE문을 이용하여 DEPT_NM 의 값을 일괄적으로 UPDATE 함

```
SELECT EMP_NO, EMP_NM, DEPT_CD, DEPT_NM
  FROM TB_EMP
  WHERE ROWNUM <= 10;
```

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM
9999999999	김회장	999999	회장실
1000000001	이경오	100001	운영본부
1000000002	이현승	100002	지원팀
1000000003	이정수	100002	지원팀
1000000004	이승준	100002	지원팀
1000000005	김현희	100002	지원팀
1000000006	김혜진	100003	기획팀
1000000007	이순자	100003	기획팀
1000000008	김려원	100003	기획팀
1000000009	박태범	100003	기획팀

```
ALTER TABLE TB_EMP DROP COLUMN DEPT_NM;
```

➢ INSERT문에 사용되는 서브쿼리

```
CREATE TABLE INSERT_SUBQUERY_TEST
(
     EMP_NO CHAR(10)
  ,  MAX_PAY_AMT NUMBER(15)
);

INSERT INTO INSERT_SUBQUERY_TEST
VALUES ('1000000001', (SELECT MAX(PAY_AMT) FROM TB_SAL_HIS WHERE EMP_NO = '1000000001'))
;

COMMIT;
```

❖ INSERT_SUBQUERY_TEST테이블을 생성하고
❖ 사원 번호가 "1000000001"인 직원의 최고 급여를 삽입함

```
SELECT * FROM INSERT_SUBQUERY_TEST;
```

EMP_NO	MAX_PAY_AMT
1000000001	5890000

```
DROP TABLE INSERT_SUBQUERY_TEST;
```

➢ 뷰 사용의 장점

장점	설명
독립성	• 테이블 구조가 변경되어도 뷰를 사용하는 응용프로그램은 변경하지 않아도 된다.
편리성	• 복잡한 질의를 뷰로 생성함으로써 관련 질의를 단순하게 작성할 수 있다. 또한 해당 형태의 SQL문을 자주 사용할 때 뷰를 이용하면 편리하게 사용할 수 있다.
보안성	• 직원의 급여 정보와 같이 숨기고 싶은 정보가 존재한다면, 뷰를 생성 할 때 해당 칼럼을 빼고 생성함으로써 사용자에게 정보를 감출 수 있다.

테이블복제
CTAS
CVAS
뷰생성

➢ 뷰 사용 실습

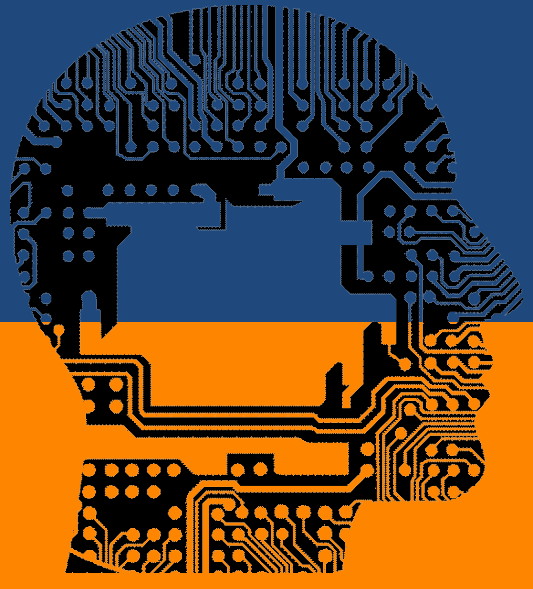
```
CREATE VIEW V_TB_SAL_HIS_MAX_BY_EMP_NO
AS
SELECT A.EMP_NO, A.EMP_NM, B.DEPT_CD, B.DEPT_NM
     , MAX(C.PAY_AMT) AS MAX_PAY_AMT
  FROM TB_EMP A , TB_DEPT B, TB_SAL_HIS C
 WHERE A.DEPT_CD = B.DEPT_CD
   AND A.EMP_NO = C.EMP_NO
GROUP BY A.EMP_NO, A.EMP_NM, B.DEPT_CD, B.DEPT_NM
;
```

```
SELECT * FROM V_TB_SAL_HIS_MAX_BY_EMP_NO;
```

```
DROP VIEW V_TB_SAL_HIS_MAX_BY_EMP_NO;
```

EMP_NO	EMP_NM	DEPT_CD	DEPT_NM	PAY_AMT
1000000003	이정수	100002	지원팀	5560000
1000000005	김현희	100002	지원팀	5860000
1000000009	박태범	100003	기획팀	5840000
1000000015	이정직	100006	데이터팀	5960000
1000000022	김순수	100007	개발팀	5690000
1000000030	이규호	100010	개발팀	5710000
1000000037	이현정	100013	빅데이터팀	5870000
1000000004	이승준	100002	지원팀	5800000
1000000019	박정혜	100007	개발팀	5980000
중간생략...				
1000000033	홍사기	100012	인공지능팀	5830000
1000000001	이경오	100001	운영본부	5890000
1000000002	이현승	100002	지원팀	5560000
1000000029	장나라	100010	개발팀	5980000
1000000034	김익정	100012	인공지능팀	5890000
1000000038	김혜수	100013	빅데이터팀	5850000
1000000039	이박력	100013	빅데이터팀	5940000

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정
DataBase

2. 그룹 함수

➢ 그룹 함수란?

① 그룹 함수를 이용하여 특정 집합의 소계, 중계, 합계, 총 합계를 구할 수 있다.
② 즉 이러한 합계를 계산하기 위해서 기존에 들어갔던 다양한 노력들이 그룹 함수를 이용하여 간단하게 처리 할 수 있게 되었다.

➢ 그룹 함수의 종류

종류	설명
ROLLUP	• 소 그룹간의 소계를 계산하는 기능 • ROLLUP함수 내에 인자로 지정된 GRUPING 칼럼은 SUBTOTAL을 생성하는데 사용된다. • GROUPING 칼럼의 수가 N이라고 했을 때 N+1의 SUBTOTAL이 생성된다. • ROLLUP함수 내의 인자의 순서가 바뀌면 결과도 바뀌게 된다.(ROLLUP은 계층 구조임)
CUBE	• 다차원적인 소계를 계산하는 기능 • 결합 가능한 모든 값에 대하여 다차원 집계를 생성 • CUBE함수 내에 칼럼이 N개라면 2의 N승만큼의 SUBTOTAL이 생성됨 • 시스템에 많은 부담을 주기때문에 사용상 주의가 필요함
GROUPING SETS	• 특정 항목에 대한 소계를 계산하는 기능

➤ 그룹 함수 실습 - 합계 데이터 출력

```
SELECT A.DEPT_CD "부서코드"
   , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.DEPT_CD) AS "부서명"
   , COUNT(*) AS "직원수"
   , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
   , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
   , (
      SELECT B.EMP_NO
           , SUM(B.PAY_AMT) AS "연봉"
        FROM TB_SAL_HIS B
       WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
       GROUP BY B.EMP_NO
       ORDER BY B.EMP_NO
     ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY A.DEPT_CD
ORDER BY A.DEPT_CD
;
```

❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함

부서코드	부서명	직원수	연봉합계	평균연봉
100001	운영본부	1	₩46,270,000	₩46,270,000
100002	지원팀	4	₩201,290,000	₩50,322,500
100003	기획팀	4	₩201,870,000	₩50,467,500
100004	디자인팀	4	₩197,450,000	₩49,362,500
100005	플랫폼사업본부	1	₩58,690,000	₩58,690,000
100006	데이터팀	4	₩204,700,000	₩51,175,000
100007	개발팀	4	₩198,190,000	₩49,547,500
100008	솔루션사업본부	1	₩46,430,000	₩46,430,000
100009	운영팀	4	₩203,040,000	₩50,760,000
100010	개발팀	4	₩199,010,000	₩49,752,500
100011	신사업본부	1	₩53,490,000	₩53,490,000
100012	인공지능팀	4	₩212,340,000	₩53,085,000
100013	빅데이터팀	4	₩211,820,000	₩52,955,000
999999	회장실	1	₩44,330,000	₩44,330,000

➢ 그룹 함수 실습 - 합계 데이터 출력 - ROLLUP함수 사용

```
SELECT A.DEPT_CD "부서코드"
   , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.DEPT_CD) AS "부서명"
   , COUNT(*) AS "직원수"
   , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
   , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
   , (
      SELECT B.EMP_NO
           , SUM(B.PAY_AMT) AS "연봉"
        FROM TB_SAL_HIS B
       WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
       GROUP BY B.EMP_NO
       ORDER BY B.EMP_NO
     ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY ROLLUP(A.DEPT_CD)
ORDER BY A.DEPT_CD ;
```

❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함
❖ ROLLUP함수를 이용하여 전체 직원수, 연봉합계, 평균연봉 까지도 출력함

부서코드	부서명	직원수	연봉합계	평균연봉
100001	운영본부	1	?46,270,000	?46,270,000
100002	지원팀	4	?201,290,000	?50,322,500
100003	기획팀	4	?201,870,000	?50,467,500
100004	디자인팀	4	?197,450,000	?49,362,500
100005	플랫폼사업본부	1	?58,690,000	?58,690,000
100006	데이터팀	4	?204,700,000	?51,175,000
100007	개발팀	4	?198,190,000	?49,547,500
100008	솔루션사업본부	1	?46,430,000	?46,430,000
100009	운영팀	4	?203,040,000	?50,760,000
100010	개발팀	4	?199,010,000	?49,752,500
100011	신사업본부	1	?53,490,000	?53,490,000
100012	인공지능팀	4	?212,340,000	?53,085,000
100013	빅데이터팀	4	?211,820,000	?52,955,000
999999	회장실	1	?44,330,000	?44,330,000
(NULL)	(NULL)	41	?2,078,920,000	?50,705,365

➢ 그룹 함수 실습 - 합계 데이터 출력 - ROLLUP함수 사용 - 인자 값 추가

```
SELECT A.DEPT_CD "부서코드"
    , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.DEPT_CD) AS "부서명"
    , A.SEX_CD AS "성별코드"
    , COUNT(*) AS "직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO
      ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY ROLLUP(A.DEPT_CD,A.SEX_CD)
ORDER BY A.DEPT_CD, A.SEX_CD
;
```

ROLLUP

- ❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함
- ❖ ROLLUP함수를 이용하여 전체 직원수, 연봉합계, 평균연봉 까지도 출력함
- ❖ ROLLUP함수를 이용하여 부서성별별 직원수, 연봉합계, 평균연봉 까지도 출력함

부서코드	부서명	성별코드	직원수	연봉합계	평균연봉
100001	운영본부	1	1	₩46,270,000	₩46,270,000
100001	운영본부	(NULL)	1	₩46,270,000	₩46,270,000
100002	지원팀	1	3	₩148,490,000	₩49,496,666
100002	지원팀	2	1	₩52,800,000	₩52,800,000
100002	지원팀	(NULL)	4	₩201,290,000	₩50,322,500
100003	기획팀	1	1	₩48,800,000	₩48,800,000
100003	기획팀	2	3	₩153,070,000	₩51,023,333
100003	기획팀	(NULL)	4	₩201,870,000	₩50,467,500
100004	디자인팀	1	1	₩48,990,000	₩48,990,000
100004	디자인팀	2	3	₩148,460,000	₩49,486,666
100004	디자인팀	(NULL)	4	₩197,450,000	₩49,362,500
100005	플랫폼사업본부	1	1	₩58,690,000	₩58,690,000
100005	플랫폼사업본부	(NULL)	1	₩58,690,000	₩58,690,000
100006	데이터팀	1	2	₩98,420,000	₩49,210,000
100006	데이터팀	2	2	₩106,280,000	₩53,140,000
100006	데이터팀	(NULL)	4	₩204,700,000	₩51,175,000
100007	개발팀	1	1	₩46,780,000	₩46,780,000
100007	개발팀	2	3	₩151,410,000	₩50,470,000
100007	개발팀	(NULL)	4	₩198,190,000	₩49,547,500
100008	솔루션사업본부	1	1	₩46,430,000	₩46,430,000
100008	솔루션사업본부	(NULL)	1	₩46,430,000	₩46,430,000
100009	운영팀	1	2	₩97,920,000	₩48,960,000
100009	운영팀	2	2	₩105,120,000	₩52,560,000
100009	운영팀	(NULL)	4	₩203,040,000	₩50,760,000
100010	개발팀	1	1	₩50,210,000	₩50,210,000
100010	개발팀	2	3	₩148,800,000	₩49,600,000
100010	개발팀	(NULL)	4	₩199,010,000	₩49,752,500
100011	신사업본부	1	1	₩53,490,000	₩53,490,000
100011	신사업본부	(NULL)	1	₩53,490,000	₩53,490,000
100012	인공지능팀	1	2	₩99,760,000	₩49,880,000
100012	인공지능팀	2	2	₩112,580,000	₩56,290,000
100012	인공지능팀	(NULL)	4	₩212,340,000	₩53,085,000
100013	빅데이터팀	1	1	₩51,000,000	₩51,000,000
100013	빅데이터팀	2	3	₩160,820,000	₩53,606,666
100013	빅데이터팀	(NULL)	4	₩211,820,000	₩52,955,000
999999	회장실	1	1	₩44,330,000	₩44,330,000
999999	회장실	(NULL)	1	₩44,330,000	₩44,330,000

전체통계(NULL)

➢ 그룹 함수 실습 - 합계 데이터 출력 - ROLLUP함수 사용 - 인자 값 추가 - GROUPING 함수

```
SELECT CASE GROUPING(A.DEPT_CD) WHEN 1 THEN '모든부서' ELSE A.DEPT_CD END AS "부서코드"
    , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.DEPT_CD) AS "부서명"
    , CASE GROUPING(A.SEX_CD) WHEN 1 THEN '모든성별' ELSE A.SEX_CD END AS "성별코드"
    , COUNT(*) AS "직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO
      ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY ROLLUP(A.DEPT_CD,A.SEX_CD)
ORDER BY A.DEPT_CD, A.SEX_CD
;
```

- ❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함
- ❖ ROLLUP함수를 이용하여 전체 직원수, 연봉합계, 평균연봉 까지도 출력함
- ❖ ROLLUP함수를 이용하여 부서성별별 직원수, 연봉합계, 평균연봉 까지도 출력함
- ❖ GROUPING 함수를 이용하여 모든성별, 모든부서라고 표기함

부서코드	부서명	성별코드	직원수	연봉합계	평균연봉
100001	운영본부	1	1	₩46,270,000	₩46,270,000
100001	운영본부	모든성별	1	₩46,270,000	₩46,270,000
100002	지원팀	1	3	₩148,490,000	₩49,496,666
100002	지원팀	2	1	₩52,800,000	₩52,800,000
100002	지원팀	모든성별	4	₩201,290,000	₩50,322,500
100003	기획팀	1	1	₩48,800,000	₩48,800,000
100003	기획팀	2	3	₩153,070,000	₩51,023,333
100003	기획팀	모든성별	4	₩201,870,000	₩50,467,500
100004	디자인팀	1	1	₩48,990,000	₩48,990,000
100004	디자인팀	2	3	₩148,460,000	₩49,486,666
100004	디자인팀	모든성별	4	₩197,450,000	₩49,362,500
100005	플랫폼사업본부	1	1	₩58,690,000	₩58,690,000
100005	플랫폼사업본부	모든성별	1	₩58,690,000	₩58,690,000
100006	데이터팀	1	2	₩98,420,000	₩49,210,000
100006	데이터팀	2	2	₩106,280,000	₩53,140,000
100006	데이터팀	모든성별	4	₩204,700,000	₩51,175,000
100007	개발팀	1	1	₩46,780,000	₩46,780,000
100007	개발팀	2	3	₩151,410,000	₩50,470,000
100007	개발팀	모든성별	4	₩198,190,000	₩49,547,500
100008	솔루션사업본부	1	1	₩46,430,000	₩46,430,000
100008	솔루션사업본부	모든성별	1	₩46,430,000	₩46,430,000
100009	운영팀	1	2	₩97,920,000	₩48,960,000
100009	운영팀	2	2	₩105,120,000	₩52,560,000
100009	운영팀	모든성별	4	₩203,040,000	₩50,760,000
100010	개발팀	1	1	₩50,210,000	₩50,210,000
100010	개발팀	2	3	₩148,800,000	₩49,600,000
100010	개발팀	모든성별	4	₩199,010,000	₩49,752,500
100011	신사업본부	1	1	₩53,490,000	₩53,490,000
100011	신사업본부	모든성별	1	₩53,490,000	₩53,490,000
100012	인공지능팀	1	2	₩99,760,000	₩49,880,000
100012	인공지능팀	2	2	₩112,580,000	₩56,290,000
100012	인공지능팀	모든성별	4	₩212,340,000	₩53,085,000
100013	빅데이터팀	1	1	₩51,000,000	₩51,000,000
100013	빅데이터팀	2	3	₩160,820,000	₩53,606,666
100013	빅데이터팀	모든성별	4	₩211,820,000	₩52,955,000
999999	회장실	1	1	₩44,330,000	₩44,330,000
999999	회장실	모든성별	1	₩44,330,000	₩44,330,000
모든부서	(NULL)	모든성별	41	₩2,078,920,000	₩50,705,365

➢ 그룹 함수 실습 - 합계 데이터 출력 - CUBE함수 사용

```
SELECT A.DEPT_CD "부서코드"
    , (SELECT L.DEPT_NM
         FROM TB_DEPT L
         WHERE L.DEPT_CD = A.DEPT_CD) AS "부서명"
    , A.SEX_CD AS "성별코드"
    , COUNT(*) AS "직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO
      ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY CUBE(A.DEPT_CD, A.SEX_CD)
ORDER BY A.DEPT_CD ;
```

- ❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함
- ❖ CUBE함수를 사용함으로써 다차원 집계를 구함
- ❖ 전체, 부서별, 부서성별별, 성별별, 합계를 구함 (CUBE함수의 인자 칼럼의 2개임, 2의 2승 =4)

부서코드	부서명	성별코드	직원수	연봉합계	평균연봉
100001	운영본부	1	1	₩46,270,000	₩46,270,000
100001	운영본부	(NULL)	1	₩46,270,000	₩46,270,000
100002	지원팀	1	3	₩148,490,000	₩49,496,666
100002	지원팀	2	1	₩52,800,000	₩52,800,000
100002	지원팀	(NULL)	4	₩201,290,000	₩50,322,500
100003	기획팀	1	1	₩48,800,000	₩48,800,000
100003	기획팀	2	3	₩153,070,000	₩51,023,333
100003	기획팀	(NULL)	4	₩201,870,000	₩50,467,500
100004	디자인팀	1	1	₩48,990,000	₩48,990,000
100004	디자인팀	2	3	₩148,460,000	₩49,486,666
100004	디자인팀	(NULL)	4	₩197,450,000	₩49,362,500
100005	플랫폼사업본부	1	1	₩58,690,000	₩58,690,000
100005	플랫폼사업본부	(NULL)	1	₩58,690,000	₩58,690,000
100006	데이터팀	1	2	₩98,420,000	₩49,210,000
100006	데이터팀	2	2	₩106,280,000	₩53,140,000
100006	데이터팀	(NULL)	4	₩204,700,000	₩51,175,000
100007	개발팀	1	1	₩46,780,000	₩46,780,000
100007	개발팀	2	3	₩151,410,000	₩50,470,000
100007	개발팀	(NULL)	4	₩198,190,000	₩49,547,500
100008	솔루션사업본부	1	1	₩46,430,000	₩46,430,000
100008	솔루션사업본부	(NULL)	1	₩46,430,000	₩46,430,000
100009	운영팀	1	2	₩97,920,000	₩48,960,000
100009	운영팀	2	2	₩105,120,000	₩52,560,000
100009	운영팀	(NULL)	4	₩203,040,000	₩50,760,000
100010	개발팀	1	1	₩50,210,000	₩50,210,000
100010	개발팀	2	3	₩148,800,000	₩49,600,000
100010	개발팀	(NULL)	4	₩199,010,000	₩49,752,500
100011	신사업본부	1	1	₩53,490,000	₩53,490,000
100011	신사업본부	(NULL)	1	₩53,490,000	₩53,490,000
100012	인공지능팀	1	2	₩99,760,000	₩49,880,000
100012	인공지능팀	2	2	₩112,580,000	₩56,290,000
100012	인공지능팀	(NULL)	4	₩212,340,000	₩53,085,000
100013	빅데이터팀	1	1	₩51,000,000	₩51,000,000
100013	빅데이터팀	2	3	₩160,820,000	₩53,606,666
100013	빅데이터팀	(NULL)	4	₩211,820,000	₩52,955,000
999999	회장실	1	1	₩44,330,000	₩44,330,000
999999	회장실	(NULL)	1	₩44,330,000	₩44,330,000
(NULL)	(NULL)	1	19	₩939,580,000	₩49,451,578
(NULL)	(NULL)	2	22	₩1,139,340,000	₩51,788,181

➢ 그룹 함수 실습 - 합계 데이터 출력 - UNION ALL & GROUP BY

```
SELECT A.DEPT_CD "부서코드", '모든성별' AS "성별코드"
    , COUNT(*) AS "직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY A.DEPT_CD
UNION ALL
SELECT '모든부서' AS "부서코드", A.SEX_CD AS "성별코드"
    , COUNT(*) AS "부서별직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "부서별연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "부서별평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY A.SEX_CD
ORDER BY "부서코드", "성별코드";
```

❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함

❖ UNION ALL 및 GROUP BY를 이용하여 부서별, 성별별 인원수와, 연봉합계, 평균연봉을 출력함

부서코드	성별코드	직원수	연봉합계	평균연봉
100001	모든성별	1	₩46,270,000	₩46,270,000
100002	모든성별	4	₩201,290,000	₩50,322,500
100003	모든성별	4	₩201,870,000	₩50,467,500
100004	모든성별	4	₩197,450,000	₩49,362,500
100005	모든성별	1	₩58,690,000	₩58,690,000
100006	모든성별	4	₩204,700,000	₩51,175,000
100007	모든성별	4	₩198,190,000	₩49,547,500
100008	모든성별	1	₩46,430,000	₩46,430,000
100009	모든성별	4	₩203,040,000	₩50,760,000
100010	모든성별	4	₩199,010,000	₩49,752,500
100011	모든성별	1	₩53,490,000	₩53,490,000
100012	모든성별	4	₩212,340,000	₩53,085,000
100013	모든성별	4	₩211,820,000	₩52,955,000
999999	모든성별	1	₩44,330,000	₩44,330,000
모든부서	1	19	₩939,580,000	₩49,451,578
모든부서	2	22	₩1,139,340,000	₩51,788,181

➢ 그룹 함수 실습 - 합계 데이터 출력 - GROUPING SETS

```
SELECT DECODE(GROUPING(A.DEPT_CD), 1, '모든부서', A.DEPT_CD) AS "부서코드"
    , DECODE(GROUPING(A.SEX_CD), 1, '모든성별', A.SEX_CD) AS "성별코드"
    , COUNT(*) AS "직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO
      ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY GROUPING SETS(A.DEPT_CD, A.SEX_CD)
ORDER BY "부서코드", "성별코드"
;
```

- ❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함
- ❖ GROUPING SETS함수를 이용하여 부서별, 성별별 인원수와, 연봉합계, 평균연봉을 출력함
- ❖ GROUPING 함수를 이용하여 모든부서, 모든성별을 출력한다.

부서코드	성별코드	직원수	연봉합계	평균연봉
100001	모든성별	1	₩46,270,000	₩46,270,000
100002	모든성별	4	₩201,290,000	₩50,322,500
100003	모든성별	4	₩201,870,000	₩50,467,500
100004	모든성별	4	₩197,450,000	₩49,362,500
100005	모든성별	1	₩58,690,000	₩58,690,000
100006	모든성별	4	₩204,700,000	₩51,175,000
100007	모든성별	4	₩198,190,000	₩49,547,500
100008	모든성별	1	₩46,430,000	₩46,430,000
100009	모든성별	4	₩203,040,000	₩50,760,000
100010	모든성별	4	₩199,010,000	₩49,752,500
100011	모든성별	1	₩53,490,000	₩53,490,000
100012	모든성별	4	₩212,340,000	₩53,085,000
100013	모든성별	4	₩211,820,000	₩52,955,000
999999	모든성별	1	₩44,330,000	₩44,330,000
모든부서	1	19	₩939,580,000	₩49,451,578
모든부서	2	22	₩1,139,340,000	₩51,788,181

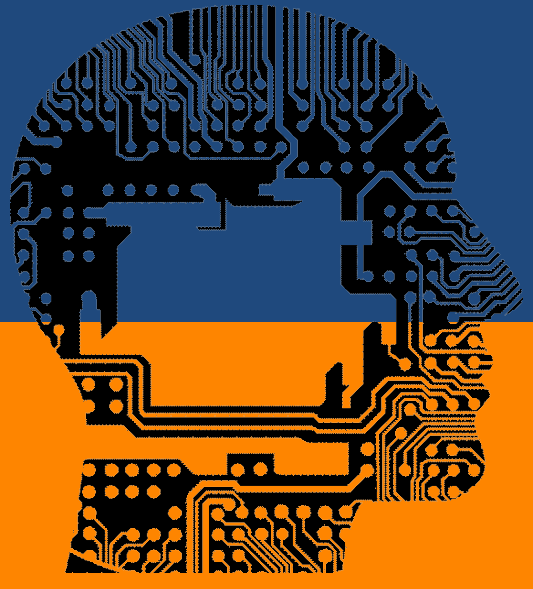
➢ 그룹 함수 실습 - 합계 데이터 출력 - GROUPING SETS

```
SELECT DECODE(GROUPING(A.DEPT_CD), 1, '모든부서', A.DEPT_CD) AS "부서코드"
    , DECODE(GROUPING(A.SEX_CD), 1, '모든성별', A.SEX_CD) AS "성별코드"
    , COUNT(*) AS "직원수"
    , TO_CHAR(TRUNC(SUM(B."연봉")), 'L999,999,999,999') AS "연봉합계"
    , TO_CHAR(TRUNC(AVG(B."연봉")), 'L999,999,999,999') AS "평균연봉"
FROM TB_EMP A
    , (
       SELECT B.EMP_NO
            , SUM(B.PAY_AMT) AS "연봉"
         FROM TB_SAL_HIS B
        WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
        GROUP BY B.EMP_NO
        ORDER BY B.EMP_NO
      ) B
WHERE A.EMP_NO = B.EMP_NO
GROUP BY GROUPING SETS(A.DEPT_CD, A.SEX_CD)
ORDER BY "부서코드", "성별코드"
;
```

- ❖ 2019년도 기준 부서별 직원수, 연봉합계, 평균연봉을 출력함
- ❖ GROUPING SETS함수를 이용하여 부서별, 성별별 인원수와, 연봉합계, 평균연봉을 출력함
- ❖ GROUPING SETS의 인자의 순서를 바꾸어도 결과는 같음

부서코드	성별코드	직원수	연봉합계	평균연봉
100001	모든성별	1	₩46,270,000	₩46,270,000
100002	모든성별	4	₩201,290,000	₩50,322,500
100003	모든성별	4	₩201,870,000	₩50,467,500
100004	모든성별	4	₩197,450,000	₩49,362,500
100005	모든성별	1	₩58,690,000	₩58,690,000
100006	모든성별	4	₩204,700,000	₩51,175,000
100007	모든성별	4	₩198,190,000	₩49,547,500
100008	모든성별	1	₩46,430,000	₩46,430,000
100009	모든성별	4	₩203,040,000	₩50,760,000
100010	모든성별	4	₩199,010,000	₩49,752,500
100011	모든성별	1	₩53,490,000	₩53,490,000
100012	모든성별	4	₩212,340,000	₩53,085,000
100013	모든성별	4	₩211,820,000	₩52,955,000
999999	모든성별	1	₩44,330,000	₩44,330,000
모든부서	1	19	₩939,580,000	₩49,451,578
모든부서	2	22	₩1,139,340,000	₩51,788,181

한국IT진흥부설
정보보호교육학원 아이섹



JAVA 웹 개발자 양성과정

DataBase

3. 윈도우 함수

➢ 윈도우 함수 개요

① 행과 행간의 관계에서 다양한 연산 처리를 할 수 있는 함수가 윈도우 함수이다.
② 분석함수로도 알려져 있다. (ANSI 표준은 윈도우 함수이다.)
③ 윈도우함수는 일반 함수와 달리 중첩하여 호출 될 수는 없다.

➢ 윈도우 함수의 종류

종류	설명
순위관련함수	• RANK • DENSE_RANK • ROW_NUMBER
집계관련함수	• SUM • MAX • MIN • AVG • COUNT
행순서관련함수	• FIRST_VALUE • LAST_VALUE • LAG • LEAD
그룹내 비율관련함수	• CUME_DIST • PERCENT_RANK • NTILE • RATIO_TO_REPORT

➢ 윈도우 함수 문법

```
SELECT
       윈도우함수(인자) OVER (PARTITION BY 칼럼 ORDER BY 칼럼)
       윈도우절
  FROM 테이블명
;
```

항목	설명
윈도우함수	• 다양한 윈도우 함수를 사용 가능
인자	• 함수에 따라 0~N개의 인자를 사용
PARTITION BY	• 전체 집합을 기준에 의해 소그룹으로 나눌 수 있다.
ORDER BY	• 어떤 항목에 대해 순위를 지정할지 ORDER BY 절을 기술할 수 있다.
윈도우절	• 함수의 대상이 되는 행 기준의 범위를 지정할 수 있다. • ROWS는 물리적인 결과 행의 수를 뜻하고 • RANGE는 논리적인 값에 의한 범위를 뜻한다.

➢ 윈도우 함수 실습 - 그룹내순위함수

```
SELECT A.EMP_NO, A.EMP_NM, A.BIRTH_DE, A.DEPT_CD
     , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.DEPT_CD) AS DEPT_NM
     , RANK() OVER(ORDER BY A.BIRTH_DE) AS RANK
     , DENSE_RANK() OVER(ORDER BY A.BIRTH_DE) AS DENSE_RANK
     , ROW_NUMBER() OVER(ORDER BY A.BIRTH_DE) AS ROW_NUMBER
     , RANK() OVER(PARTITION BY A.DEPT_CD ORDER BY A.BIRTH_DE) AS RANK_DEPT_CD
  FROM TB_EMP A
  WHERE A.SEX_CD = '1' --남성
  ORDER BY A.BIRTH_DE    ;
```

- ❖ 전 직원중 성별이 남성이 직원들의 생년월일을 출력하고 생년월일 순으로 RANK를 구함
- ❖ RANK 함수는 동일값이라면 동일 순위라고 판단함 1 2 2 4 순으로 순위를 정함
- ❖ DENSE_RANK 함수는 동일값이라면 동일 순위라고 판단함 1 2 2 3 순으로 순위를 정함
- ❖ ROW_NUMBER 함수는 동일값이라도 무조건 순위를 정함
- ❖ DEPT_CD기준으로 PARTITION BY하여 부서별 생일 순위도 같이 구하였음

EMP_NO	EMP_NM	BIRTH_DE	DEPT_CD	DEPT_NM	RANK	DENSE_RANK	ROW_NUMBER	RANK_DEPT_CD
9999999999	김회장	19651105	999999	회장실	1	1	1	1
1000000026	김길정	19690524	100009	운영팀	2	2	2	1
1000000035	최창수	19690524	100012	인공지능팀	2	2	3	1
1000000032	이준표	19771202	100011	신사업본부	4	3	4	1
1000000014	이관심	19780213	100005	플랫폼사업본부	5	4	5	1
1000000023	이관심	19780213	100008	솔루션사업본부	5	4	6	1
1000000001	이경오	19840612	100001	운영본부	7	5	7	1
1000000024	박선영	19870615	100009	운영팀	8	6	8	2
1000000033	홍사기	19870615	100012	인공지능팀	8	6	9	2
1000000004	이승준	19870623	100002	지원팀	10	7	10	1
1000000017	이겸손	19880124	100006	데이터팀	11	8	11	1
1000000009	박태범	19880629	100003	기획팀	12	9	12	1
1000000002	이현승	19880705	100002	지원팀	13	10	13	2
1000000012	김호형	19910227	100004	디자인팀	14	11	14	1
1000000021	김열호	19910227	100007	개발팀	14	11	15	1
1000000003	이정수	19911224	100002	지원팀	16	12	16	3
1000000039	이박력	19940709	100013	빅데이터팀	17	13	17	1
1000000030	이규호	19970627	100010	개발팀	18	14	18	1
1000000015	이정직	19980715	100006	데이터팀	19	15	19	2

➢ 윈도우 함수 실습 - 집계관련함수

```
SELECT A.EMP_NO
     , A.MAX_EMP_NM
     , A.연봉
     , A.MAX_DEPT_CD
     , (SELECT DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.MAX_DEPT_CD) AS DEPT_NM
     , SUM(A.연봉) OVER(PARTITION BY A.MAX_DEPT_CD) AS "속한부서의연봉총액"
     , SUM(A.연봉) OVER(PARTITION BY A.MAX_DEPT_CD ORDER BY A.연봉
                        RANGE UNBOUNDED PRECEDING) AS "속한부서의연봉누적합계"
     , MAX(A.연봉) OVER(PARTITION BY A.MAX_DEPT_CD) AS "속한부서의최고연봉액"
     , MIN(A.연봉) OVER(PARTITION BY A.MAX_DEPT_CD) AS "속한부서의최저연봉액"
     , TRUNC(AVG(A.연봉) OVER(PARTITION BY A.MAX_DEPT_CD)) AS "속한부서의평균연봉액"
     , TRUNC(AVG(A.연봉) OVER(PARTITION BY A.MAX_DEPT_CD ORDER BY A.연봉
                              ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING
                             )
             ) AS "속한부서에서앞뒤자신의평균연봉액"
     , COUNT(*) OVER (PARTITION BY A.MAX_DEPT_CD) AS "부서별직원수"
  FROM
     (
     SELECT B.EMP_NO
          , MAX(A.EMP_NM) AS MAX_EMP_NM
          , MAX(A.DEPT_CD) AS MAX_DEPT_CD
          , SUM(B.PAY_AMT) AS "연봉"
       FROM TB_SAL_HIS B , TB_EMP A
      WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
      AND A.EMP_NO = B.EMP_NO
      GROUP BY B.EMP_NO
      ORDER BY B.EMP_NO
     ) A
ORDER BY A.MAX_DEPT_CD, A.연봉;
```

- ❖ 2019년 기준 전직원의 연봉액수를 구하고
- ❖ 자신이 속한 부서의 연봉총액이 얼마인지를 구함
- ❖ 자신이 속한 부서의 연봉총액이 얼마인지를 구하는데 누적합계로 보여줌 (RANGE UNBOUNDED PRECEDINGD은 현재행을 기준으로 파티션내 첫 번째 행까지의 범위를 지정함)
- ❖ 자신이 속한 부서의 최고연봉액수를 보여줌
- ❖ 자신이 속한 부서의 최저연봉액수를 보여줌
- ❖ 자신이 속한 부서의 평균연봉액수를 보여줌
- ❖ 자신이 속한 부서의 앞자신뒤의평균연봉액수를 보여줌
- ❖ 자신이 속한 부서의 직원수를보여줌

EMP_NO	MAX_EMP_NM	연봉	MAX_DEPT_CD	DEPT_NM	속한부서의연봉총액	속한부서의연봉누적합계	속한부서의최고연봉액	속한부서의최저연봉액	속한부서의평균연봉액	속한부서에서앞뒤자신의평균연봉액	부서별직원수
1000000001	이경오	46270000	100001	운영본부	46270000	46270000	46270000	46270000	46270000	46270000	1
1000000003	이정수	45740000	100002	지원팀	201290000	45740000	53530000	45740000	50322500	47480000	4
1000000004	이승준	49220000	100002	지원팀	201290000	94960000	53530000	45740000	50322500	49253333	4
1000000005	김현희	52800000	100002	지원팀	201290000	147760000	53530000	45740000	50322500	51850000	4
1000000002	이현승	53530000	100002	지원팀	201290000	201290000	53530000	45740000	50322500	53165000	4
1000000007	이순자	47760000	100003	기획팀	201870000	47760000	53620000	47760000	50467500	48280000	4
1000000009	박태범	48800000	100003	기획팀	201870000	96560000	53620000	47760000	50467500	49416666	4
1000000008	김려원	51690000	100003	기획팀	201870000	148250000	53620000	47760000	50467500	51370000	4
1000000006	김혜진	53620000	100003	기획팀	201870000	201870000	53620000	47760000	50467500	52655000	4
1000000013	이나라	46740000	100004	디자인팀	197450000	46740000	52400000	46740000	49362500	47865000	4
1000000012	김호형	48990000	100004	디자인팀	197450000	95730000	52400000	46740000	49362500	48350000	4
1000000010	박혜령	49320000	100004	디자인팀	197450000	145050000	52400000	46740000	49362500	50236666	4
1000000011	최수자	52400000	100004	디자인팀	197450000	197450000	52400000	46740000	49362500	50860000	4
1000000014	이관심	58690000	100005	플랫폼사업본부	58690000	58690000	58690000	58690000	58690000	58690000	1
1000000015	이정직	49180000	100006	데이터팀	204700000	49180000	54700000	49180000	51175000	49210000	4
1000000017	이겸손	49240000	100006	데이터팀	204700000	98420000	54700000	49180000	51175000	50000000	4
1000000018	박바른	51580000	100006	데이터팀	204700000	150000000	54700000	49180000	51175000	51840000	4
1000000016	이진실	54700000	100006	데이터팀	204700000	204700000	54700000	49180000	51175000	53140000	4
1000000021	김열호	46780000	100007	개발팀	198190000	46780000	55270000	46780000	49547500	46835000	4
1000000020	최정진	46890000	100007	개발팀	198190000	93670000	55270000	46780000	49547500	47640000	4
1000000022	김순수	49250000	100007	개발팀	198190000	142920000	55270000	46780000	49547500	50470000	4
1000000019	박정혜	55270000	100007	개발팀	198190000	198190000	55270000	46780000	49547500	52260000	4
1000000023	이관심	46430000	100008	솔루션사업본부	46430000	46430000	46430000	46430000	46430000	46430000	1
1000000024	박선영	47140000	100009	운영팀	203040000	47140000	56220000	47140000	50760000	48020000	4
1000000027	박이수	48900000	100009	운영팀	203040000	96040000	56220000	47140000	50760000	48940000	4
1000000026	김길정	50780000	100009	운영팀	203040000	146820000	56220000	47140000	50760000	51966666	4
1000000025	박호진	56220000	100009	운영팀	203040000	203040000	56220000	47140000	50760000	53500000	4
1000000029	장나라	44380000	100010	개발팀	199010000	44380000	54460000	44380000	49752500	47170000	4
1000000031	김사랑	49960000	100010	개발팀	199010000	94340000	54460000	44380000	49752500	48183333	4
1000000030	이규호	50210000	100010	개발팀	199010000	144550000	54460000	44380000	49752500	51543333	4
1000000028	김나라	54460000	100010	개발팀	199010000	199010000	54460000	44380000	49752500	52335000	4
1000000032	이준표	53490000	100011	신사업본부	53490000	53490000	53490000	53490000	53490000	53490000	1
1000000035	최창수	49090000	100012	인공지능팀	212340000	49090000	57190000	49090000	53085000	49880000	4
1000000033	홍사기	50670000	100012	인공지능팀	212340000	99760000	57190000	49090000	53085000	51716666	4
1000000034	김익정	55390000	100012	인공지능팀	212340000	155150000	57190000	49090000	53085000	54416666	4
1000000036	박여진	57190000	100012	인공지능팀	212340000	212340000	57190000	49090000	53085000	56290000	4
1000000039	이박력	51000000	100013	빅데이터팀	211820000	51000000	54220000	51000000	52955000	51850000	4
1000000037	이현정	52700000	100013	빅데이터팀	211820000	103700000	54220000	51000000	52955000	52533333	4
1000000038	김혜수	53900000	100013	빅데이터팀	211820000	157600000	54220000	51000000	52955000	53606666	4
1000000040	김여진	54220000	100013	빅데이터팀	211820000	211820000	54220000	51000000	52955000	54060000	4
9999999999	김회장	44330000	999999	회장실	44330000	44330000	44330000	44330000	44330000	44330000	1

➢ 윈도우 함수 실습 - 행순서관련함수 - 실습 환경 구축

```
DROP TABLE TB_REAL_IDX PURGE;

CREATE TABLE TB_REAL_IDX
(
    SEQ NUMBER(15)
    , SECTOR_NM VARCHAR2(50)
    , STD_DE CHAR(8)
    , STD_TM CHAR(6)
    , CUR_IDX NUMBER(15, 2)
    , CONSTRAINT PK_TB_REAL_IDX
     PRIMARY KEY(SEQ)
)
;
```

```
INSERT INTO TB_REAL_IDX
SELECT ROWNUM AS RNUM
     , '코스피' AS SECTOR_NM
     , '20200629' AS STD_DE
     , TO_CHAR(TO_DATE('090000', 'HH24MISS') + (ROWNUM*60)/24/60/60, 'HH24MISS') AS HH24MISS
     , CUR_IDX
  FROM
  (
  SELECT
       ROUND(DBMS_RANDOM.VALUE(2000.00, 2050.99), 2) AS CUR_IDX
    FROM DUAL CONNECT BY LEVEL <= 390
    ORDER BY CUR_IDX
  )
UNION ALL
SELECT ROWNUM+390 AS RNUM
     , '코스닥' AS SECTOR_NM
     , '20200629' AS STD_DE
     , TO_CHAR(TO_DATE('090000', 'HH24MISS') + (ROWNUM*60)/24/60/60, 'HH24MISS') AS HH24MISS
     , CUR_IDX
  FROM
  (
  SELECT
       ROUND(DBMS_RANDOM.VALUE(700.00, 725.99), 2) AS CUR_IDX
    FROM DUAL CONNECT BY LEVEL <= 390
    ORDER BY CUR_IDX
  )
;

COMMIT;
```

```
SELECT * FROM TB_REAL_IDX ORDER BY SEQ;
```

➢ 윈도우 함수 실습 - 행순서관련함수

```
SELECT A.SEQ
     , A.SECTOR_NM
     , A.STD_DE
     , A.STD_TM
     , A.CUR_IDX
     , FIRST_VALUE(CUR_IDX) OVER(PARTITION BY A.SECTOR_NM ORDER BY A.STD_TM
                                 ROWS UNBOUNDED PRECEDING) AS "각지수의첫지수값"
     , LAST_VALUE(CUR_IDX) OVER(PARTITION BY A.SECTOR_NM ORDER BY A.STD_TM
                                ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING) AS "각지수의마지막지수값"
     , LAG(CUR_IDX, 1) OVER (PARTITION BY A.SECTOR_NM ORDER BY A.STD_TM) AS "이전시간의지수값"
     , LEAD(CUR_IDX, 1) OVER (PARTITION BY A.SECTOR_NM ORDER BY A.STD_TM) AS "다음시간의지수값"
  FROM TB_REAL_IDX A
ORDER BY A.SECTOR_NM DESC, A.STD_DE, A.STD_TM
;
```

- ❖ ROWS UNBOUNDED PRECEDING : 현재행을 기준으로 파티션내의 첫번째 행까지의 범위 지정
- ❖ ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING : 현재행을 기준으로 파티션내의 마지막 행까지의 범위 지정

SEQ	SECTOR_NM	STD_DE	STD_TM	CUR_IDX	각지수의첫지수값	각지수의마지막지수값	이전시간의지수값	다음시간의지수값
1	코스피	20200629	90100	2000.11	2000.11	2050.72	(NULL)	2000.29
2	코스피	20200629	90200	2000.29	2000.11	2050.72	2000.11	2000.31
3	코스피	20200629	90300	2000.31	2000.11	2050.72	2000.29	2000.57
중간생략...								
388	코스피	20200629	152800	2050.55	2000.11	2050.72	2050.36	2050.57
389	코스피	20200629	152900	2050.57	2000.11	2050.72	2050.55	2050.72
390	코스피	20200629	153000	2050.72	2000.11	2050.72	2050.57	(NULL)
391	코스닥	20200629	90100	700.02	700.02	725.81	(NULL)	700.04
392	코스닥	20200629	90200	700.04	700.02	725.81	700.02	700.18
393	코스닥	20200629	90300	700.18	700.02	725.81	700.04	700.2
394	코스닥	20200629	90400	700.2	700.02	725.81	700.18	700.26
중간생략...								
395	코스닥	20200629	90500	700.26	700.02	725.81	700.2	700.31
777	코스닥	20200629	152700	725.57	700.02	725.81	725.57	725.58
778	코스닥	20200629	152800	725.58	700.02	725.81	725.57	725.71
779	코스닥	20200629	152900	725.71	700.02	725.81	725.58	725.81
780	코스닥	20200629	153000	725.81	700.02	725.81	725.71	(NULL)

➢ 윈도우 함수 실습 - 그룹내 비율관련함수

```
SELECT
       A.MAX_DEPT_CD
     , A.부서별연봉총액
     , (SELECT L.DEPT_NM FROM TB_DEPT L WHERE L.DEPT_CD = A.MAX_DEPT_CD) AS DEPT_NM
     , ROUND(RATIO_TO_REPORT(A.부서별연봉총액) OVER(), 4) * 100 || '%' AS "부서별연봉비율"
     , ROUND(PERCENT_RANK() OVER(ORDER BY A.부서별연봉총액), 4) *100 || '%' AS "부서별연봉비율순서별백분율"
     , ROUND(CUME_DIST() OVER(ORDER BY A.부서별연봉총액), 4) *100 || '%' AS "부서별연봉비율순서별누적백분율"
     , NTILE(4) OVER(ORDER BY A.부서별연봉총액) AS "부서별연봉비율순서별등분결과"
FROM
(
SELECT
       A.MAX_DEPT_CD
     , SUM(A.연봉) AS "부서별연봉총액"
  FROM
     (
     SELECT B.EMP_NO
          , MAX(A.EMP_NM) AS MAX_EMP_NM
          , MAX(A.DEPT_CD) AS MAX_DEPT_CD
          , SUM(B.PAY_AMT) AS "연봉"
       FROM TB_SAL_HIS B , TB_EMP A
      WHERE B.PAY_DE  BETWEEN '20190101' AND '20191231'
      AND A.EMP_NO = B.EMP_NO
      GROUP BY B.EMP_NO
      ORDER BY B.EMP_NO
     ) A
GROUP BY A.MAX_DEPT_CD
ORDER BY A.MAX_DEPT_CD
) A
 ;
```

- ❖ RATIO_TO_REPORT 함수로 부서별 연봉의 비율을 구함
- ❖ PERCENT_RANK함수로 부서별연봉비율순서의 백분율을 구함
- ❖ CUME_DIST함수로 부서별연봉비율의 누적 백분율을 구함
- ❖ NTILE함수로 부서별연봉비율의 등분 결과를 구함

MAX_DEPT_CD	부서별연봉총액	DEPT_NM	부서별연봉비율	부서별연봉비율순서별백분율	부서별연봉비율순서별누적백분율	부서별연봉비율순서별등분결과
999999	44330000	회장실	2.13%	0%	7.14%	1
100001	46270000	운영본부	2.23%	7.69%	14.29%	1
100008	46430000	솔루션사업본부	2.23%	15.38%	21.43%	1
100011	53490000	신사업본부	2.57%	23.08%	28.57%	1
100005	58690000	플랫폼사업본부	2.82%	30.77%	35.71%	2
100004	197450000	디자인팀	9.50%	38.46%	42.86%	2
100007	198190000	개발팀	9.53%	46.15%	50%	2
100010	199010000	개발팀	9.57%	53.85%	57.14%	2
100002	201290000	지원팀	9.68%	61.54%	64.29%	3
100003	201870000	기획팀	9.71%	69.23%	71.43%	3
100009	203040000	운영팀	9.77%	76.92%	78.57%	3
100006	204700000	데이터팀	9.85%	84.62%	85.71%	4
100013	211820000	빅데이터팀	10.19%	92.31%	92.86%	4
100012	212340000	인공지능팀	10.21%	100%	100%	4

감사합니다
THANK YOU