GROUP BY
& HAVING

▶ ORDER BY

SELECT한 컬럼에 대해 정렬을 할 때 작성하는 구문으로
SELECT 구문의 가장 마지막에 작성하며 실행 순서 역시
가장 마지막에 수행됨

✓ 표현식

SELECT 컬럼 명 [, 컬럼명, ...]
FROM 테이블 명
WHERE 조건식
ORDER BY 컬럼명 | 별칭 | 컬럼 순번 정렬방식 [NULLS FIRST | LAST];

* 정렬 방식
- ASC : 오름차순
- DESC : 내림차순

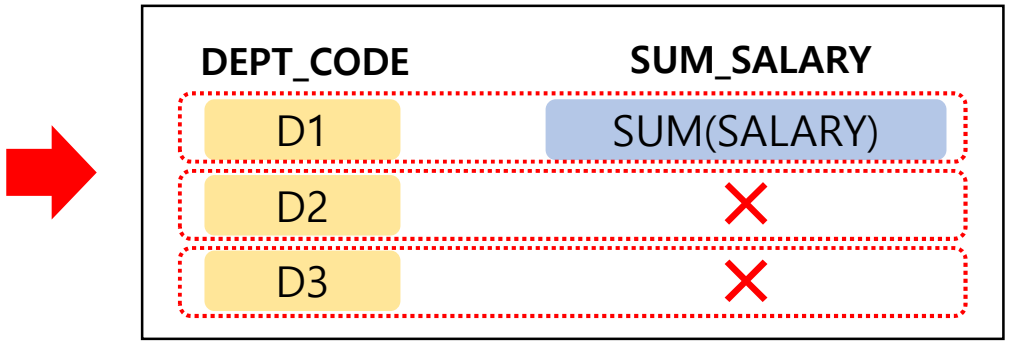
▶ GROUP BY

그룹 함수는 단 한 개의 결과 값만 산출하기 때문에 그룹이 여러 개일 경우 오류 발생

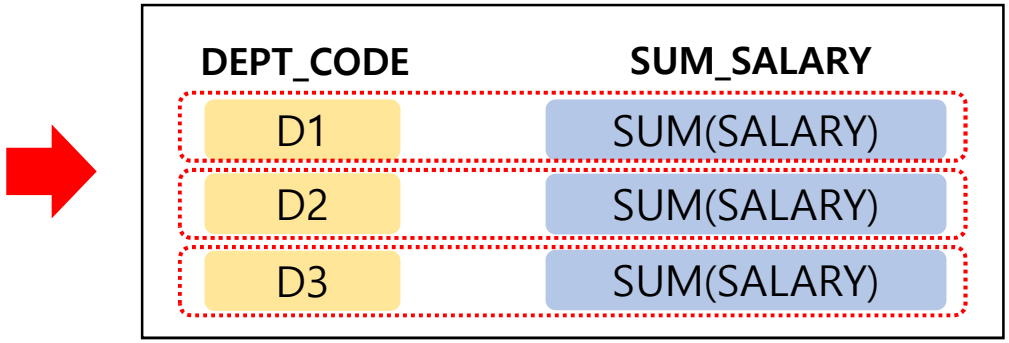
여러 개의 결과 값을 산출하기 위해 그룹 함수가 적용될 그룹의 기준을 GROUP BY절에 기술하여 사용

```
SELECT DEPT_CODE,
       SUM(SALARY)
FROM EMPLOYEE;
```

** 에러 발생




```
SELECT DEPT_CODE,
       SUM(SALARY)
FROM EMPLOYEE
GROUP BY DEPT_CODE;
```



▶ GROUP BY

✓ 예시

- **EMPLOYEE에서 부서코드, 그룹 별 급여의 합계, 그룹 별 급여의 평균(정수처리), 인원 수를 조회하고 부서 코드 순으로 정렬**

```
SELECT DEPT_CODE 부서코드,
       SUM(SALARY) 합계,
       FLOOR(AVG(SALARY)) 평균,
       COUNT(*) 인원수
FROM EMPLOYEE
GROUP BY DEPT_CODE
ORDER BY DEPT_CODE ASC;
```

	DEPT_CODE	합계	평균	인원수
1	D1	7820000	2606666	3
2	D2	6520000	2173333	3
3	D5	15760000	2626666	6
4	D6	10100000	3366666	3
5	D8	6986240	2328746	3
6	D9	17700000	5900000	3
7	(null)	5210000	2605000	2

- **EMPLOYEE테이블에서 부서코드와 보너스 받는 사원 수 조회하고 부서코드 순으로 정렬**

```
SELECT DEPT_CODE 부서코드,
       COUNT(BONUS) 인원수
FROM EMPLOYEE
WHERE BONUS IS NOT NULL
GROUP BY DEPT_CODE
ORDER BY DEPT_CODE ASC;
```

	DEPT_CODE	COUNT(BONUS)
1	D1	2
2	D5	2
3	D6	1
4	D8	2
5	D9	1
6	(null)	1

▶ GROUP BY

✓ 예시

- EMPLOYEE테이블에서 성별과 성별 별 급여 평균(정수처리), 급여 합계, 인원 수 조회하고 인원수로 내림차순 정렬

```
SELECT DECODE(SUBSTR(EMP_NO, 8, 1), 1, '남', 2, '여') 성별,
       FLOOR(AVG(SALARY)) 평균,
       SUM(SALARY) 합계,
       COUNT(*) 인원수
FROM EMPLOYEE
GROUP BY DECODE(SUBSTR(EMP_NO, 8, 1), 1, '남', 2, '여')
ORDER BY 4 DESC;
```

	성별	평균	합계	인원수
1	남	3317333	49760000	15
2	여	2542030	20336240	8

▶ HAVING

그룹 함수로 값을 구해올 그룹에 대해 조건을 설정할 때 HAVING절에 기술
(WHERE절은 SELECT에 대한 조건)

✓ 예시

- **부서 코드와 급여 3000000 이상인 직원의 그룹별 평균 조회**

```
SELECT DEPT_CODE, FLOOR(AVG(SALARY)) 평균
FROM EMPLOYEE
WHERE SALARY >= 3000000
GROUP BY DEPT_CODE
ORDER BY 1;
```

- **부서 코드와 급여 평균이 3000000 이상인 그룹 조회**

```
SELECT DEPT_CODE, FLOOR(AVG(SALARY)) 평균
FROM EMPLOYEE
GROUP BY DEPT_CODE
HAVING FLOOR(AVG(SALARY)) >= 3000000
ORDER BY DEPT_CODE;
```

▶ ROLLUP과 CUBE

그룹 별 산출한 결과 값의 집계를 계산하는 함수

✓ 예시

```
SELECT JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY ROLLUP(JOB_CODE)
ORDER BY 1;
```

	JOB_CODE	SUM(SALARY)
1	J1	8000000
2	J2	9700000
3	J3	10800000
4	J4	9320000
5	J5	8460000
6	J6	15746240
7	J7	8070000
8	(null)	70096240

```
SELECT JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY CUBE(JOB_CODE)
ORDER BY 1;
```

▶ ROLLUP과 CUBE

✓ ROLLUP

인자로 전달받은 그룹 중 가장 먼저 지정한 그룹별로
추가적 집계 결과 반환

✓ 예시

```
SELECT DEPT_CODE, JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY ROLLUP(DEPT_CODE, JOB_CODE)
ORDER BY 1;
```

	DEPT_CODE	JOB_CODE	SUM(SALARY)
1	D1	J6	6440000
2	D1	J7	1380000
3	D1	(null)	7820000
4	D2	J4	6520000
5	D2	(null)	6520000
6	D5	J3	3500000
7	D5	J5	8460000
8	D5	J7	3800000
9	D5	(null)	15760000
10	D6	J3	7300000
11	D6	J4	2800000
12	D6	(null)	10100000
13	D8	J6	6986240
14	D8	(null)	6986240
15	D9	J1	8000000
16	D9	J2	9700000
17	D9	(null)	17700000
18	(null)	J6	2320000
19	(null)	J7	2890000
20	(null)	(null)	5210000
21	(null)	(null)	70096240

▶ ROLLUP과 CUBE

✓ CUBE

인자로 지정된 그룹들로 가능한 모든 조합 별로 집계한 결과 반환

✓ 예시

```
SELECT DEPT_CODE, JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY CUBE(DEPT_CODE, JOB_CODE)
ORDER BY 1;
```

	DEPT_CODE	JOB_CODE	SUM(SALARY)
1	D1	J6	6440000
2	D1	J7	1380000
3	D1	(null)	7820000
4	D2	J4	6520000
5	D2	(null)	6520000
6	D5	J3	3500000
7	D5	J5	8460000
8	D5	J7	3800000
9	D5	(null)	15760000
10	D6	J3	7300000
11	D6	J4	2800000
12	D6	(null)	10100000
13	D8	J6	6986240
14	D8	(null)	6986240
15	D9	J1	8000000
16	D9	J2	9700000
17	D9	(null)	17700000
18	(null)	J1	8000000
19	(null)	J2	9700000
20	(null)	J3	10800000
21	(null)	J4	9320000
22	(null)	J5	8460000
23	(null)	J6	2320000
24	(null)	J6	15746240
25	(null)	J7	2890000
26	(null)	J7	8070000
27	(null)	(null)	5210000
28	(null)	(null)	70096240

ROLLUP과 CUBE

```
SELECT DEPT_CODE, JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY ROLLUP(DEPT_CODE, JOB_CODE)
UNION
SELECT '', JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY ROLLUP(JOB_CODE)
ORDER BY 1;
```

```
SELECT DEPT_CODE, JOB_CODE, SUM(SALARY)
FROM EMPLOYEE
GROUP BY CUBE(DEPT_CODE, JOB_CODE)
ORDER BY 1;
```

▶ ROLLUP과 CUBE

✓ GROUPING

ROLLUP이나 CUBE에 의한 집계 산출물이
인자로 전달받은 컬럼 집합의
산출물이면 0 반환
아니면 1 반환

	DEPT_CODE	JOB_CODE	SUM(SALARY)	구분
1	D1	J6	6440000	그룹별 합계
2	D1	J7	1380000	그룹별 합계
3	D1	(null)	7820000	부서별 합계
4	D2	J4	6520000	그룹별 합계
5	D2	(null)	6520000	부서별 합계
6	D5	J3	3500000	그룹별 합계
7	D5	J5	8460000	그룹별 합계
8	D5	J7	3800000	그룹별 합계
9	D5	(null)	15760000	부서별 합계
10	D6	J3	7300000	그룹별 합계
11	D6	J4	2800000	그룹별 합계
12	D6	(null)	10100000	부서별 합계
13	D8	J6	6986240	그룹별 합계
14	D8	(null)	6986240	부서별 합계
15	D9	J1	8000000	그룹별 합계
16	D9	J2	9700000	그룹별 합계
17	D9	(null)	17700000	부서별 합계
18	(null)	J1	8000000	직급별 합계
19	(null)	J2	9700000	직급별 합계
20	(null)	J3	10800000	직급별 합계
21	(null)	J4	9320000	직급별 합계
22	(null)	J5	8460000	직급별 합계
23	(null)	J6	2320000	그룹별 합계
24	(null)	J6	15746240	직급별 합계
25	(null)	J7	2890000	그룹별 합계
26	(null)	J7	8070000	직급별 합계
27	(null)	(null)	5210000	부서별 합계
28	(null)	(null)	70096240	총 합계

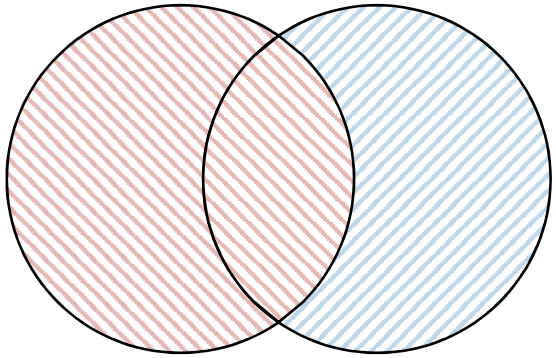
▶ ROLLUP과 CUBE

✓ GROUPING 예시

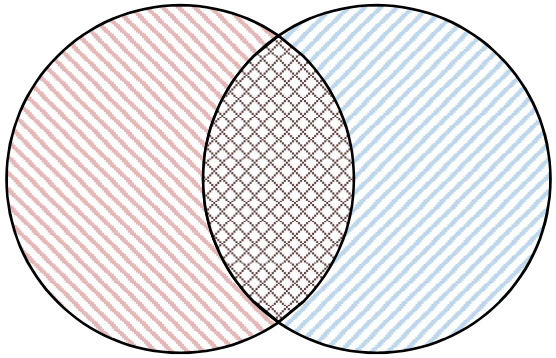
```
SELECT DEPT_CODE, JOB_CODE, SUM(SALARY),
        CASE WHEN GROUPING(DEPT_CODE) = 0 AND
                  GROUPING(JOB_CODE) = 1
             THEN '부서별 합계'
             WHEN GROUPING(DEPT_CODE) = 1 AND
                  GROUPING(JOB_CODE) = 0
             THEN '직급별 합계'
             WHEN GROUPING(DEPT_CODE) = 1 AND
                  GROUPING(JOB_CODE) = 1
             THEN '총 합계'
             ELSE '그룹별 합계'
        END AS 구분
FROM EMPLOYEE
GROUP BY CUBE(DEPT_CODE, JOB_CODE)
ORDER BY 1;
```

▶ 집합 연산자

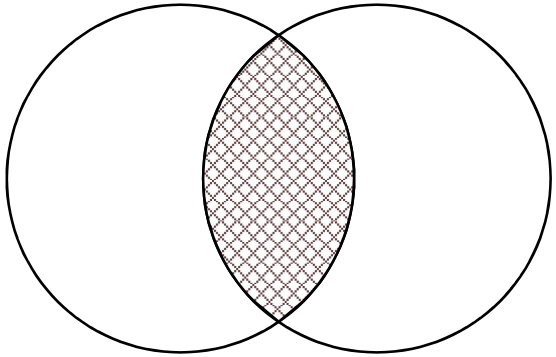
여러 개의 SELECT 결과물을 하나의 쿼리로 만드는 연산자



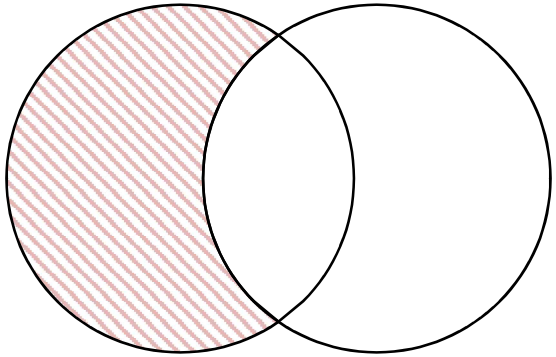
UNION



UNION ALL



INTERSECT



MINUS

▶ 집합 연산자

✓ UNION

여러 개의 쿼리 결과를 합치는 연산자로 중복된 영역은 제외하여 합침

✓ 예시

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5'

UNION

SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	200	선동일	D9	8000000
2	201	송중기	D9	6000000
3	202	노옹철	D9	3700000
4	204	유재식	D6	3400000
5	205	정중하	D6	3900000
6	206	박나라	D5	1800000
7	207	하이유	D5	2200000
8	208	김해술	D5	2500000
9	209	심봉선	D5	3500000
10	210	윤은해	D5	2000000
11	215	대북혼	D5	3760000
12	217	전지연	D1	3660000

▶ 집합 연산자

✓ INTERSECT

여러 개의 SELECT 결과에서 공통된 부분만 결과로 추출(교집합)

✓ 예시

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5'
```

INTERSECT

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	209	심봉선	D5	3500000
2	215	대북혼	D5	3760000

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE SALARY > 3000000;
```

▶ 집합 연산자

✓ UNION ALL

여러 쿼리 결과를 합치는 연산자로 중복된 영역 모두 포함하여 합침

✓ 예시

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5'

UNION ALL

SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	206	박나라	D5	1800000
2	207	하이유	D5	2200000
3	208	김해술	D5	2500000
4	209	심봉선	D5	3500000
5	210	윤은해	D5	2000000
6	215	대북혼	D5	3760000
7	200	선동일	D9	8000000
8	201	송종기	D9	6000000
9	202	노옹철	D9	3700000
10	204	유재식	D6	3400000
11	205	정중하	D6	3900000
12	209	심봉선	D5	3500000
13	215	대북혼	D5	3760000
14	217	전지연	D1	3660000

▶ 집합 연산자

✓ MINUS

선행 SELECT 결과에서 다음 SELECT 결과와 겹치는 부분을 제외한 나머지 부분 추출(차집합)

✓ 예시

```
SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE DEPT_CODE = 'D5'

MINUS

SELECT EMP_ID, EMP_NAME, DEPT_CODE, SALARY
FROM EMPLOYEE
WHERE SALARY > 3000000;
```

	EMP_ID	EMP_NAME	DEPT_CODE	SALARY
1	206	박나라	D5	1800000
2	207	하이유	D5	2200000
3	208	김해술	D5	2500000
4	210	윤은해	D5	2000000

▶ GROUPING SETS

그룹 별로 처리된 여러 개의 SELECT문을 하나로 합친 결과를 원할 때 사용
(집합 연산자 사용과 동일)

✓ 예시

```
SELECT DEPT_CODE, JOB_CODE, MANAGER_ID, FLOOR(AVG(SALARY))
FROM EMPLOYEE
GROUP BY GROUPING SETS((DEPT_CODE, JOB, MANAGER_ID),
                       (DEPT_CODE, MANAGER_ID),
                       (JOB_CODE, MANAGER_ID));
```

	DEPT_CODE	JOB_CODE	MANAGER_ID	FLOOR(AVG(SALARY))
1	D5	J5	207	2500000
2	D6	J4	204	2800000
3	D5	J3	207	3500000
4	D9	J2	200	6000000
5	D6	J3	200	3400000
6	D8	J6	211	2550000
7	(null)	J7	(null)	2890000
8	D8	J6	100	2436240
9	(null)	J6	(null)	2320000
10	D1	J6	214	3220000
11	D6	J3	204	3900000
12	D5	J7	207	1900000
13	D5	J5	200	2200000
14	D1	J7	200	1380000
15	D2	J4	(null)	2173333

51	(null)	J5	207	2500000
52	(null)	J5	(null)	3760000
53	(null)	J6	214	3220000