# 05강 그룹 함수

내장 함수

# 그룹함수의 개념

- 그룹함수란 전체 데이터에서 통계적인 결과를 얻기 위해서 행의 집합에 적용하여 하나의 결과를 만들어 내는 함수를 의미한다.

| 그룹함수 | 설명 |
|---|---|
| SUM() | 총 합계를 구한다. |
| AVG() | 평균을 구한다. |
| MAX() | 최대값을 구한다 |
| MIN() | 최소값을 구한다. |
| COUNT() | 행의 개수를 센다 |
| COUNT(Distinct) | 행의 개수를 센다(중복 부분은 1개만 인정) |
| STDEV() | 표준편차를 구한다 |
| VARIANCE() | 분산을 구한다. |

- 그룹함수는 대부분 숫자형에 적용하지만 MAX와 MIN은 그외 모든 데이터 유형에서 사용할 수 있다

# 그룹함수의 개념

```
SELECT SUM(salary) AS "급여 총액",
       AVG(salary) AS "급여 평균",
       MAX(salary) AS "최대 급여",
       MIN(salary) AS "최소 급여"
FROM employee;
```

```
SELECT MAX(hiredate),
       MIN(hiredate)
FROM employee;
```

# 그룹함수와 NULL

- NULL은 기본적으로 연산이 불가능하다
- 그룹함수는 NULL값을 연산에서 제외하고 연산을 진행한다.

```
SELECT SUM(commission)AS "커미션 총액"
FROM employee;
```

# 행의 개수 구하는 COUNT 함수

- COUNT(*) 모든 행의 개수를 구한다.

```
SELECT COUNT(*) AS "사원 수"
FROM employee;
```

- COUNT(컬럼) : 해당 컬럼에서 NULL을 제외하고 개수를 구한다.

```
SELECT SUM(commission)AS "커미션 총액"
FROM employee;

SELECT COUNT(commission) AS "커미션을 받는 사원 수"
FROM employee
WHERE commission IS NOT NULL;
```

# 행의 개수 구하는 COUNT 함수

- DISTINCT를 사용해서 중복된 값을 제거한 개수를 구할 수 있다

```
SELECT job
FROM employee
ORDER BY job ASC;
```

```
SELECT COUNT(job) AS "직업의 수"
FROM employee;
```

```
SELECT COUNT(DISTINCT job) AS "직업의 종류"
FROM employee;
```

# 그룹함수와 일반 컬럼

- 그룹함수의 결과는 기본적으로 1개이다.

- SELECT 에서 그룹함수와 일반 컬럼을 같이 조회하게 되면
하나의 결과와 여러 결과를 같이 보여 주어야 하는데 이때 오류가 발생한다.

```
SELECT ename, MAX(salary)
FROM employee;
```

# 데이터 그룹 => GROUP BY

- 기존 그룹함수는 테이블당 하나의 결과값만을 구하는 용도로 사용됨
- 그러나 특정 컬럼을 기준로 그룹을 지어서 그룹별 결과를 구할 필요가 있다.
- 이때 그룹을 지어주는 구문이 GROUP BY 이다

GROUP BY 그룹 지어줄 컬럼.

```
SELECT AVG(salary) AS "급여 평균"
FROM employee
GROUP BY dno;
```

# 데이터 그룹 => GROUP BY

- 기존에는 그룹함수와 일반 컬럼을 같이 사용할 수 없지만
일반 컬럼을 그룹화 한 경우 그룹함수와 같이 사용이 가능하다.

오히려 같이 사용하면 그룹함수의 결과가 어떤 그룹의 결과인지를 보여주므로 그룹화한 컬럼과 그룹함수를 묶어서 사용하는 것이 좋다.

```
SELECT dno AS "부서번호", AVG(salary) AS "급여 평균"
FROM employee
GROUP BY dno;
```

다만 그룹화 하지 않은 컬럼과 같이 사용하면 오류가 발생한다.

```
SELECT dno,ename, AVG(salary) AS "급여 평균"
FROM employee
GROUP BY dno;
```

# 데이터 그룹 => GROUP BY

- GROUP BY 절에 두개 이상의 컬럼명을 작성하면 상위 그룹과 하위그룹으로 나뉘어져 결과를 반환한다.

```
SELECT dno,job, SUM(salary)
FROM employee
GROUP BY dno,job
ORDER BY dno ASC, job ASC;
```

# 그룹 결과의 제한

- 다음과 같은 상황을 가정하자
- 업무별 그룹화된 평균 급여가 2000이상인 그룹의 급여 총액을 구해보고자 한다
- 조건이 있기 때문에 다음과 같이 쿼리문을 작성하겠지만 오류가 발생한다.

```
SELECT job, SUM(salary)
FROM employee
WHERE avg(salary)>=2000
GROUP BY job;
```

- 그룹함수의 결과에 조건을 주고 싶다면 GROUP BY절 뒤에 HAVING 절을 주어야 한다.

```
SELECT job,avg(salary), SUM(salary)
FROM employee
GROUP BY job
HAVING avg(salary)>=2000;
```

# 그룹 결과의 제한

- 예제 : 부서별 최고 급여가 3000 이상인 부서의 번호와 부서별 최고 급여 구하기

```
SELECT dno, MAX(salary)
FROM employee
GROUP BY dno
HAVING MAX(salary) >=3000
ORDER BY dno ASC;
```

- 예제: 매니저를 제외하고 급여 총액이 5000 이상인 담당 업무(job)별 급여 총액과 해당 인원을 구하기

```
SELECT job, COUNT(*), SUM(salary)
FROM employee
WHERE job NOT LIKE '%MANAGER%'
GROUP BY job
HAVING SUM(salary)>=5000
ORDER BY SUM(salary);
```

- 예제: 부서별 평균 급여중 최고 평균 급여를 조회하기

```
SELECT MAX(AVG(salary))
FROM employee
GROUP BY dno;
```

# 추가 함수

- 단순히 전체의 합을 구하는 것이 아닌 중간 합계가 필요하다면 추가로 처리가 필요하게 된다.
  - 이때 사용하는 하는 함수는 ROLLUP()이다.

```
SELECT job, SUM(salary)
FROM employee
GROUP BY ROLLUP(job);
```

  - rollup의 컬럼을 기준으로 소연산을 진행한다.

  - 컬럼이 여러 개라면 컬럼 순서대로 그룹지어서 소연산을 진행한다.

```
SELECT job, dno, SUM(salary)
FROM employee
GROUP BY ROLLUP(job,dno);
```