suanlab

데이터베이스
Database

ORACLE®

DATABASE

08
집계 및 그룹 함수

```
COUNT(), SUM(), AVG(), MAX(), MIN()
GROUP BY
HAVING
```

집계 함수

- 여러 행에 대해 하나의 결과를 출력하는 그룹 함수를 이용하여 여러가지 집계 연산을 수행
- 집계 함수 종류

함수	설명	예
COUNT()	행의 개수	COUNT(salary)
SUM()	합계	SUM(salary)
AVG()	평균	AVG(salary)
MIN()	최솟값	MIN(salary)
MAX()	최댓값	MAX(salary)
STDDEV()	표준편차	STDDEV(salary)
VARIANCE()	분산	VARIANCE(salary)

- COUNT()
  - 열의 행 개수를 구하는 함수

```
SELECT COUNT(salary)
FROM employees;
```

```
SELECT COUNT(manager_id)
FROM employees;
```

```
SELECT COUNT(commission_pct)
FROM employees;
```

```
SELECT COUNT(*)
FROM employees;
```

집계 함수

- SUM() / AVG()
  - 열의 합계를 구하는 SUM() 함수, 열의 평균을 구하는 AVG() 함수

```
SELECT SUM(salary), AVG(salary)
FROM employees;
```

```
SELECT SUM(salary) / COUNT(salary)
FROM employees;
```

```
SELECT first_name, salary,
       SUM(salary) OVER (ORDER BY first_name)
FROM employees;
```

- MIN() / MAX()
  - 열의 최솟값을 구하는 MIN() 함수, 열의 최댓값을 구하는 MAX() 함수

```
SELECT MIN(salary), MAX(salary)
FROM employees;
```

```
SELECT MIN(first_name), MAX(first_name)
FROM employees;
```

- STDDEV() / VARIANCE()
  - 표준편차를 구하는 STDDEV() 함수, 분산을 구하는 VARIANCE() 함수

```
SELECT STDDEV(salary), VARIANCE(salary)
FROM employees;
```

```
SELECT first_name, salary,
       STDDEV(salary) OVER (ORDER BY first_name)
FROM employees
WHERE department_id = 50;
```

GROUP BY

- 지정한 열의 데이터 값을 기준으로 그룹화하여 집계 함수 적용
- GROUP BY 동작 순서
  - 테이블에서 WHERE 조건식에 맞는 데이터 값만 구분
  - 지정한 열 기준으로 같은 데이터 값으로 그룹화
  - 지정한 열들의 그룹화된 집계 결과 출력
- GROUP BY 절 특징
  - WHERE 절은 그룹화 되기 전에 조건식 적용
  - GROUP BY 절 사용시 SELECT 절에 지정된 기준 열을 지정
  - SELECT 절에 그룹 함수 없이도 GROUP BY 절 사용 가능

```
SELECT job_id, SUM(salary), AVG(salary)
FROM employees
GROUP BY job_id;
```

```
SELECT job_id, SUM(salary), AVG(salary)
FROM employees
WHERE department_id = 50
GROUP BY job_id;
```

```
SELECT department_id, MIN(salary), MAX(salary)
FROM employees
GROUP BY department_id;
```

```
SELECT department_id, MIN(salary), MAX(salary)
FROM employees
WHERE hire_date > '20070101'
GROUP BY department_id;
```

```
SELECT country_id, COUNT(country_id)
FROM locations
GROUP BY country_id
ORDER BY country_id;
```

GROUP BY

- 다중 GROUP BY 절

```
SELECT job_id, department_id,
       SUM(salary), AVG(salary)
FROM employees
WHERE department_id BETWEEN 50 AND 100
GROUP BY job_id, department_id
ORDER BY job_id;
```

```
SELECT department_id, manager_id,
       SUM(salary), AVG(salary)
FROM employees
WHERE department_id = 50
GROUP BY department_id, manager_id
ORDER BY manager_id;
```

```
SELECT manager_id, department_id, job_id,
       SUM(salary), MIN(salary), MAX(salary)
FROM employees
WHERE manager_id IN (100, 101)
GROUP BY manager_id, department_id, job_id
ORDER BY manager_id, department_id
```

HAVING

- WHERE 절에서는 그룹 함수를 사용할 수 없음
- 그룹화된 집계 결과에 조건식을 적용할 때 HAVING 절 사용

```
SELECT job_id, SUM(salary), AVG(salary)
FROM employees
GROUP BY job_id
HAVING AVG(salary) > 10000;
```

```
SELECT department_id, MIN(salary), MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary) > 7000;
```

```
SELECT country_id, COUNT(country_id)
FROM locations
GROUP BY country_id
HAVING COUNT(country_id) > 2
ORDER BY country_id;
```

```
SELECT job_id, department_id,
       SUM(salary), AVG(salary)
FROM employees
WHERE department_id BETWEEN 50 AND 100
GROUP BY job_id, department_id
HAVING AVG(salary) > 9000
ORDER BY job_id;
```

```
SELECT manager_id, department_id, job_id,
       SUM(salary), MIN(salary), MAX(salary)
FROM employees
WHERE manager_id IN (100, 101)
GROUP BY manager_id, department_id, job_id
HAVING SUM(salary) BETWEEN 10000 AND 40000
ORDER BY manager_id, department_id;
```

[실습] 집계 함수

- employees 테이블에서 salary가 8000이상인 직원의 수를 조회

- employees 테이블에서 hire_date가 2007년 1월 1일 이후인 직원의 수를 조회

- jobs 테이블에서 max_salary 값의 합계와 평균을 조회

- employees 테이블에서 job_id가 'IT_PROG'인 직원의 salary 합계와 평균을 조회

- employees 테이블에서 department_id가 50과 80 사이인 직원의 first_name, salary, 그리고 commission_pct의 평균값을 first_name 정렬 기준으로 조회 (null 값은 0으로 출력)

[실습] 집계 함수

- jobs 테이블에서 max_salary 값의 최솟값과 max_salary 값의 최댓값을 조회

- jobs 테이블에서 job_title이 'Programmer'인 직업의 max_salary 값의 최솟값과 max_salary 값의 최댓값을 조회

- employees 테이블에서 department_id가 50인 데이터의 hire_date 최소값과 최댓값 조회

- employees 테이블에서 department_id가 100인 데이터의 first_name, salary, 그리고 salary의 분산값을 hire_date 정렬 기준으로 조회

[실습] 그룹 함수

- employees 테이블에서 hire_date 값이 2004년 1월 1일부터 2006년 12월 31일 사이의 데이터를 job_id 기준으로 그룹화한 뒤에 job_id와 salary 최솟값과 최대값을 조회

- employees 테이블에서 department_id 가 50과 80인 데이터를 department_id와 job_id 기준으로 그룹화한 뒤에 department_id와 job_id, salary 합계, 최솟값, 최대값을 job_id 기준으로 정렬하여 조회

- employees 테이블에서 department_id와 job_id 기준으로 그룹화한 뒤에 salary 평균값이 12000 이상인 데이터만 department_id와 job_id, salary 최솟값, 최대값, 평균을 department_id 기준으로 정렬하여 조회


suanlab
이수안 컴퓨터 연구소
suan computer laboratory