

database-9

☰ 태그	
📅 날짜	@2023년 6월 13일

select 결과를 그룹지어서 보고 싶거나, 정렬해서 보고싶거나, 통계를 내서 보고 싶을 때 사용할 수 있는 문법을 설명

내용

- ORDER BY 로 정렬해서 select하기
- aggregate function으로 통계 결과 뽑기
- GROUP BY로 그룹화 하기
- HAVING 키워드로 그룹 필터링 하기

order by

- 조회 결과를 특정 attribute(s) 기준으로 정렬하여 가져오고 싶을 때 사용
- default 정렬 방식은 오름차순이다.
- 오름차순 정렬은 ASC, 내림차순 정렬은 DESC로 표기한다.

임직원들의 정보를 연봉 순서대로 정렬해서 알고 싶다.

```
SELECT * FROM employee ORDER BY salary;
```

id	name	...	salary	dept_id
3	JENNY	...	50000000	1003
8	JOHN	...	65000000	1003
14	SAM	...	70000000	1003
11	SUZANNE	...	75000000	1005
7	MINA	...	80000000	1004
...
13	JISUNG	...	90000000	1005
1	MESSI	...	100000000	1003
4	BROWN	...	120000000	1001
5	DINGYO	...	120000000	1003
6	JULIA	...	120000000	1002
16	MUSK	...	150000000	NULL

```
SELECT * FROM employee ORDER BY salary DESC;
```

id	name	...	salary	dept_id
16	MUSK	...	150000000	NULL
4	BROWN	...	120000000	1001
5	DINGYO	...	120000000	1003
6	JULIA	...	120000000	1002
1	MESSI	...	100000000	1003
13	JISUNG	...	90000000	1005
...
7	MINA	...	80000000	1004
11	SUZANNE	...	75000000	1005
14	SAM	...	70000000	1003
8	JOHN	...	65000000	1003
3	JENNY	...	50000000	1003

```
SELECT * FROM employee ORDER BY dept_id ASC, salary DESC;
```

id	name	...	salary	dept_id
16	MUSK	...	150000000	NULL
15	SIMON	...	100000000	NULL
4	BROWN	...	120000000	1001
6	JULIA	...	120000000	1002
9	HENRY	...	82000000	1002
5	DINGYO	...	120000000	1003
1	MESSI	...	100000000	1003
10	NICOLE	...	90000000	1003
14	SAM	...	70000000	1003
8	JOHN	...	65000000	1003
3	JENNY	...	50000000	1003
...

aggregate function

- 여러 tuple들의 정보를 요약해서 하나의 값으로 추출하는 함수
- 대표적으로 count, sum max, min, avg 함수가 있다.
- (주로) 관심있는 attribute에 사용된다. ex) avg(salary) max(birth_date)
- null 값들은 제외하고 요약값을 추출

임직원 수를 알고 싶다.

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

16

(*) : 튜블의 수를 의미

```
SELECT COUNT(position) FROM employee;
```

16

같은 결과 출력

```
SELECT COUNT(dept_id) FROM employee;
```

14

dept_id가 null인 경우가 있다.

null은 count를 하지 않는다.

프로젝트 2002에 참여한 임직원 수와 최대 연봉과 최소 연봉과 평균 연봉을 알고 싶다.

```
SELECT COUNT(*), MAX(salary), MIN(salary), AVG(salary)
FROM works_on W JOIN employee E ON W.empl_id = E.id
WHERE W.proj_id = 2002;
```

```
+-----+-----+-----+-----+
| count(*) | max(salary) | min(salary) | avg(salary) |
+-----+-----+-----+-----+
|      4 | 120000000 | 70000000 | 95000000.0000 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

GROUP BY

```
SELECT W.proj_id, COUNT(*), MAX(salary), MIN(salary), AVG(salary)
FROM works_on W JOIN employee E ON W.empl_id = E.id
GROUP BY W.proj_id;
```

```
+-----+-----+-----+-----+-----+
| proj_id | count(*) | max(salary) | min(salary) | avg(salary) |
+-----+-----+-----+-----+-----+
| 2001 |      10 | 120000000 | 50000000 | 84000000.0000 |
| 2002 |       4 | 120000000 | 70000000 | 95000000.0000 |
| 2003 |       5 | 90000000 | 75000000 | 84000000.0000 |
| 2004 |       4 | 90000000 | 50000000 | 70000000.0000 |
+-----+-----+-----+-----+-----+
```

- 관심있는 attribute(s) 기준으로 그룹을 나눠서 그룹별로 aggregate function을 적용하고 싶을 때 사용
- grouping attribute(s) : 그룹을 나누는 기준이 되는 attribute(s)
- grouping attribute(s)에 null 값이 있을 때는 null 값을 가지는 tuple끼리 묶인다.

HAVING : 결과를 기준으로 어떤 조건을 걸고 싶을때 사용

프로젝트 참여 인원이 7명 이상인 프로젝트 들에 대해서 각 프로젝트에 참여한 임직원 수와 최대 연봉과 최소 연봉과 평균 연봉을 알고 싶다.

```
SELECT w.proj_id, COUNT(*), MAX(salary), MIN(salary), AVG(salary)
FROM works_on W JOIN employee E ON W.empl_id = E.id
GROUP BY W.proj_id
HAVING COUNT(*) >= 7;
```

proj_id	count(*)	max(salary)	min(salary)	avg(salary)
2001	10	120000000	50000000	84000000.0000

- GROUP BY 와 함께 사용한다.
- aggregate function 의 결과값을 바탕으로 그룹을 필터링하고 하고 싶을 때 사용
- HAVING 절에 명시된 조건을 만족하는 그룹만 결과에 포함된다.

각 부서별 인원수를 인원 수가 많은 순서대로 정렬해서 알고 싶다.

```
SELECT dept_id, COUNT(*) AS empl_count FROM employee
GROUP BY dept_id
ORDER BY empl_count DESC;
```

dept_id	empl_count
1003	6
1005	3
NULL	2
1002	2
1004	2
1001	1

SELECT

6. SELECT attribute(s) or aggregate function(s)

1. FROM table(s)

2. [WHERE condition(s)]

3. [GROUP BY group attribute(s)]

4. [HAVING group condition(s)]

5. [ORDER BY attribute(s)]

- select 쿼리에서 각 절(phrase)의 실행 순서는 개념적인 순서이다.
- select 쿼리의 실제 실행 순서는 각 RDBMS에서 어떻게 구현했는지에 따라 다르다.