SQL 집약과 자르기 – 집합의 세계

- SQL의 특징적인 사고방식 중에, **레코드** 단위가 아닌 **레코드의 '집합'** 단위로 처리를 기술하는 것이 있다.
- 이런 사용방식을 **집합 지향**이라고 부른다.
- **GROUP BY, HAVING** 구와 이것과 함께 사용하는 **SUM 또는 COUNT 등의 집약 함수**의 사용이 대표적인 것들이다.

1. 집약

- 집약 함수 : COUNT, SUM, AVG, MAX, MIN
- 이 외에 집약함수들도 있지만, 표준 SQL에 있는 집약함수는 5개뿐 이다.

1) 여러 개의 레코드를 한 개의 레코드로 집약

- 집약의 결과를 살펴보기 위해 간단한 예제를 살펴본다.
- data_1 ~ 6 필드는 사람에 대해서 무언가를 나타내는 정보다.
- data_type이 A면 data_1, data_2 // B면 data_3, data_4, data_5 // C면 data_6 의 데이터를 사용한다.

```
-- 비집약 테이블
select * from NonAggTbl;
```

	ID	DATA_TYPE	DATA_1	DATA_2	DATA_3	DATA_4	DATA_5	DATA_6
1	지용운	A	100	10	34	346	54	(null)
2	지용운	B	45	2	167	77	90	157
3	지용운	C	(null)	3	687	1355	324	457
4	김수정	A	78	5	724	457	(null)	1
5	김수정	B	123	12	178	346	85	235
6	김수정	C	45	(null)	23	46	687	33
7	이경호	A	75	0	190	25	356	(null)
8	이경호	B	435	0	183	(null)	4	325
9	이경호	C	96	128	(null)	0	0	12

```
--  문제] 위와 같이 한 사람에 대한 필요한 정보가 여러 개의 레코드에 분산되어 있을 때,
다음의 테이블과 같이 사람마다 필요한 정보만 집약한 테이블을 만들어라.
```

	ID	DATA_1	DATA_2	DATA_3	DATA_4	DATA_5	DATA_6
1	김수정	78	5	178	346	85	33
2	이경호	75	0	183	(null)	4	12
3	지용운	100	10	167	77	90	457

```
-- 정답
select id,
    max(case when data_type='A' then data_1 else null end) as data_1,
    max(case when data_type='A' then data_2 else null end) as data_2,
    max(case when data_type='B' then data_3 else null end) as data_3,
    max(case when data_type='B' then data_4 else null end) as data_4,
    max(case when data_type='B' then data_5 else null end) as data_5,
    max(case when data_type='C' then data_6 else null end) as data_6
from NonAggTbl
group by id;
```

- CASE 식과 GROUP BY 식을 응용

- GROUP BY 구로 집약했을 때, SELECT 구에 입력할 수 있는 것은 3가지다.

  - 상수
  - GROUP BY 구에서 사용한 집약 키
  - 집약함수

```
-- 위에서 사용한 쿼리 (집약 전)
select id,
    case when data_type='A' then data_1 else null end as data_1,
    case when data_type='A' then data_2 else null end as data_2,
    case when data_type='B' then data_3 else null end as data_3,
    case when data_type='B' then data_4 else null end as data_4,
    case when data_type='B' then data_5 else null end as data_5,
    case when data_type='C' then data_6 else null end as data_6
from NonAggTbl;
```

	ID	DATA_1	DATA_2	DATA_3	DATA_4	DATA_5	DATA_6
1	지용운	100	10	(null)	(null)	(null)	(null)
2	지용운	(null)	(null)	167	77	90	(null)
3	지용운	(null)	(null)	(null)	(null)	(null)	457
4	김수정	78	5	(null)	(null)	(null)	(null)
5	김수정	(null)	(null)	178	346	85	(null)
6	김수정	(null)	(null)	(null)	(null)	(null)	33
7	이경호	75	0	(null)	(null)	(null)	(null)
8	이경호	(null)	(null)	183	(null)	4	(null)
9	이경호	(null)	(null)	(null)	(null)	(null)	12

- GROUP BY로 데이터를 자르는 시점에는 각 집합에 3개의 요소가 있다.
- 그런데 여기서 집약함수가 적용되면 NULL을 제외하고 하나의 요소만 있는 집합이 만들어 진다.

2) 합쳐서 하나

문제 1

```
-- 제품의 대상 연령별 가격을 관리하는 테이블.
select * from PriceByAge;
```

	PRODUCT_ID	LOW_AGE	HIGH_AGE	PRICE
1	제품1	0	50	2000
2	제품1	51	100	3000
3	제품2	0	100	4200
4	제품3	0	20	500
5	제품3	31	70	800
6	제품3	71	100	1000
7	제품4	0	99	8900

```
-- 문제 1] 이런 제품 중 0~100 세까지 모든 연령이 가지고 놀 수 있는 제품을 구해라.
-- 아래와 같은 테이블이 나와야 한다.
-- 제품 3 의 경우, 21~30 세 연령이 빠지므로 모든 연령에 해당 X.
```

	PRODUCT_ID
1	제품1
2	제품2

```
-- 정답]
select product_id
from PriceByAge
group by product_id
having sum(high_age - low_age +1 ) = 101;
```

- 일단 집약 단위가 제품이므로 집약 키는 제품 ID로 한다.
- 이어서 각 범위에 있는 상수 개수를 모두 더한 합계가 101인 제품을 선택한다.
- 0~100까지 이므로 사이에 있는 상수의 개수는 101개라는 것에 주의!
- HAVING 구의 'high_age - low_age +1'로 각 레코드의 연령 범위에 있는 정수 개수를 구한다.
-

```
-- 참고, 정답에서 사용한 각 필드
select product_id, sum(1), sum(low_age), sum(high_age),  sum(high_age -
low_age + 1)
from PriceByAge
group by product_id;
```

	PRODUCT_ID	SUM(1)	SUM(LOW_AGE)	SUM(HIGH_AGE)	SUM(HIGH_AGE-LOW_AGE+1)
1	제품1	2	51	150	101
2	제품2	1	0	100	101
3	제품3	3	102	190	91
4	제품4	1	0	99	100

문제 2

```
-- 문제 2] 아래 테이블에서 사람들이 숙박한 날이 10 일 이상인 방을 선택한다.
-- 숙박한 날의 수는 도착일이 2 월 1 일, 출발일이 2 월 6 일이라면 5 박이므로 5 일 이다.
select * from HotelRooms;
```

	ROOM_NBR	START_DATE	END_DATE
1	101	08/02/01	08/02/06
2	101	08/02/06	08/02/08
3	101	08/02/10	08/02/13
4	202	08/02/05	08/02/08
5	202	08/02/08	08/02/11
6	202	08/02/11	08/02/12
7	303	08/02/03	08/02/17

```
-- 정답
select room_nbr, sum(end_date - start_date) as working_days
from HotelRooms
group by room_nbr
having sum(end_date - start_date) >=10;
```

	ROOM_NBR	WORKING_DAYS
1	101	10
2	303	14

2. 자르기

- 지금까지 **GROUP BY** 구에 대해 레코드의 '**집약**'이라는 측면을 강조했다.
- '**자르기**' 라는 중요한 기능이 하나 더 있다.
- "한 개의 구에 두 개의 연산이 들어 있다는 것은 GROUP BY 구에 대한 이해를 막는 원인이 된다."

1) 자르기와 파티션

 # 문제 1

```
-- 문제 1] 이름 첫 글자를 사용해 특정한 글자로 시작하는 이름을 가진 사람이 몇 명인지 집계해라.
select * from Persons;
```

	NAME	AGE	HEIGHT	WEIGHT
1	지용운	30	188	90
2	김수정	21	167	55
3	최현웅	87	158	48
4	이경호	54	187	70
5	송미정	39	177	120
6	이동건	90	175	48
7	김중수	12	160	55
8	한장희	25	182	90
9	김현창	30	176	53

```
-- 정답
select  substr(name, 1, 1) as label , count(*)
from Persons
group by substr(name, 1, 1);
```

	LABEL	COUNT(*)
1	김	3
2	이	2
3	최	1
4	지	1
5	송	1
6	한	1

문제 1-2

```
--문제 1-2] 나이를 기준으로 어린이(20 세미만), 성인(20~69 세), 노인(70 세 이상)으로
나눠라
-- 출력: 나이구분, 명수
```

	AGE_CLASS	COUNT(*)
1	노인	2
2	성인	6
3	어린이	1

```
-- 정답]
select  case  when age<20 then '어린이'
              when age between 20 and 69 then '성인'
              when age>=70 then '노인'
              else null  end as age_class,
    count(*)
from persons
group by  case  when age<20 then '어린이'
             when age between 20 and 69 then '성인'
              when age>=70 then '노인'
              else null end;
```

- 자르기의 기준이 되는 키를 GROUP BY 구와 SELECT 구 모두에 입력하는 것이 포인트.
- PostgreSQL, MySQL 에서는 SELECT 구에 붙인 'age_class'라는 별칭을 사용해 'GROUP BY age_class'처럼 단순하게 작성할 수도 있다.

문제 1-3

```
-- 문제 1-3] BMI 수치를 바탕으로 저체중(18.5 미만), 정상(18.5 이상 25 미만), 과체중(25
이상)으로 분류해라.
-- BMI  = 체중 / { (키/100)^2 }
-- 출력: 이름, BMI, 분류
select name, round(weight/power(height/100, 2), 1) as BMI,
    case when weight/power(height/100, 2) <18.5 then '저체중'
        when weight/power(height/100, 2)>=18.5 and weight/power(height/100, 2)<25
            then '정상'
        when weight/power(height/100, 2) >=25 then '과체중' else null end as 분류
from persons;
```

	NAME	BMI	분류
1	지용문	25.5	과체중
2	김수정	19.7	정상
3	최현웅	19.2	정상
4	이경호	20	정상
5	송미정	38.3	과체중
6	이동건	15.7	저체중
7	김중수	21.5	정상
8	한장희	27.2	과체중
9	김현창	17.1	저체중

 # 문제 1-4

```
-- 문제 1-4] 저체중, 정상, 과체중으로 분류하고 명수를 구하라
-- 출력: 체중분류, 명수
select
    case when weight/power(height/100, 2) <18.5 then '저체중'
        when weight/power(height/100, 2)>=18.5 and weight/power(height/100, 2)<25
then '정상'
         when weight/power(height/100, 2) >=25 then '과체중' else null end as 분류,
         count(*)
from persons
group by case when weight/power(height/100, 2) <18.5 then '저체중'
        when weight/power(height/100, 2)>=18.5 and weight/power(height/100, 2)<25
then '정상'
         when weight/power(height/100, 2) >=25 then '과체중' else null end;
```

	분류	COUNT(*)
1	저체중	2
2	정상	4
3	과체중	3

2) PARTITION BY 구를 사용한 자르기

- 'GROUP BY' 구에서 **집약 기능을 제외하고 자르는 기능만 남긴 것이** 윈도우 함수의 '**PARTITION BY**' 구.
- 실제로 집약이라는 기능을 제외하면 두 구문은 실질적인 기능에 차이가 없다.
- 한마디로 PARTITION BY 구를 사용해도 단순히 필드 이름뿐만 아니라 CASE식, 계산식을 사용한 **복잡한 기준을 사용할 수 있다는 말** 이다.

```
-- 문제] 이전에 살펴 봤던 연령 범위 테이블(Persons)에 파티션 자르기를 사용해보자.
-- 출력: 이름, 나이, age_class, age_class 별 나이 등수
```

	NAME	AGE	AGE_CLASS	AGE_RANK_IN_CLASS
1	최현웅	87	노인	1
2	이동건	90	노인	2
3	김수정	21	성인	1
4	한장희	25	성인	2
5	지용운	30	성인	3
6	김현창	30	성인	3
7	송미정	39	성인	5
8	이경호	54	성인	6
9	김중수	12	어린이	1

```
-- 정답]
select name,
    age,
    case when age<20 then '어린이'
        when age between 20 and 69 then '성인'
        when age>=70 then '노인'
        else null end as age_class,
    rank() over(partition by
        case when age<20 then '어린이'
            when age between 20 and 69 then '성인'
            when age>=70 then '노인'
            else null end
        order by age) as age_rank_in_class
from persons
order by age_class, age_rank_in_class;
```

- 마지막에 있는 age_rank_in_class가 각 파티션 내부에서의 나이 순위를 나타내는 필드.
- PARTITION BY 구는 GROUP BY 구와 달리 집약 기능이 없으므로, 원래 Persons 테이블의 레코드가 모두 원래 형태로 나오는 것을 주목.
- PARTITION BY 구는 입력에 정보를 추가할 뿐이므로 원본 테이블에 정보를 완전히 유지한다.