

sql 독학 강의 # select를 잘 이용하는 방법(2), 3편 -sTricky

sTricky 2020. 3. 19. 13:45

sql 독학 강의 # select를 잘 이용하는 방법(2), 3편 -sTricky



컨텐츠 index

1. 산술 연산자 사용해보기
2. where절에 비교 연산자를 사용해 보기

- 3. order by 절을 사용 하여 정렬하여 출력 하기
- 4. 집합 연산자 사용하기

안녕하세요.

<sql 독학 강의> 가 3번째 시간,

select를 잘 이용하는 방법으로는 두 번째 시간입니다.

지난 시간에 데이터베이스에서 사용하는 용어 정리, DESC, Projection과 Selection 개념을 이용해서 원하는 데이터만 select 하기, where 절 사용하기, 별칭(alias) 사용하기, distinct 명령어 사용하기, 문자열 합치는 함수 concat 사용하기에 대해서 배워 보았습니다.

#지난 독학 강의 보러 가기#

[2020/03/19 - \[Database/sql 강의\] - sql 공부 강의 # select를 잘 이용하는 방법\(1\), 2편 -sTricky](#)

| | |
|--|---|
| | <p>sql 공부 강의 # select를 잘 이용하는 방법(1), 2편 -sTricky</p> <p>stricky.tistory.com</p> |
|--|---|

연습은 많이들 해보셨는지 모르겠습니다. 계속 이야기드리지만 진짜 실제로 실습을 많이 해봐야지 실력이 늡니다!! 파이팅!!!

그럼 이번 시간에도 한번 열심히 달려보도록 하겠습니다.

sql 공부 강의 시작하겠습니다.

1. 산술 연산자 사용해보기

산술 연산자란 우리가 알고 있는 **+, -, *, /** 를 의미합니다. SQL에서도 다른 프로그래밍 언어와 마찬가지로 산술 연산자를 사용해서 어떤 결과를 얻을 수 있습니다.

기존에 있는 데이터로는 산술 연산자 실습이 어려우니 아래와 같이 테이블을 만들고, 데이터를 입력하도록 하겠습니다.

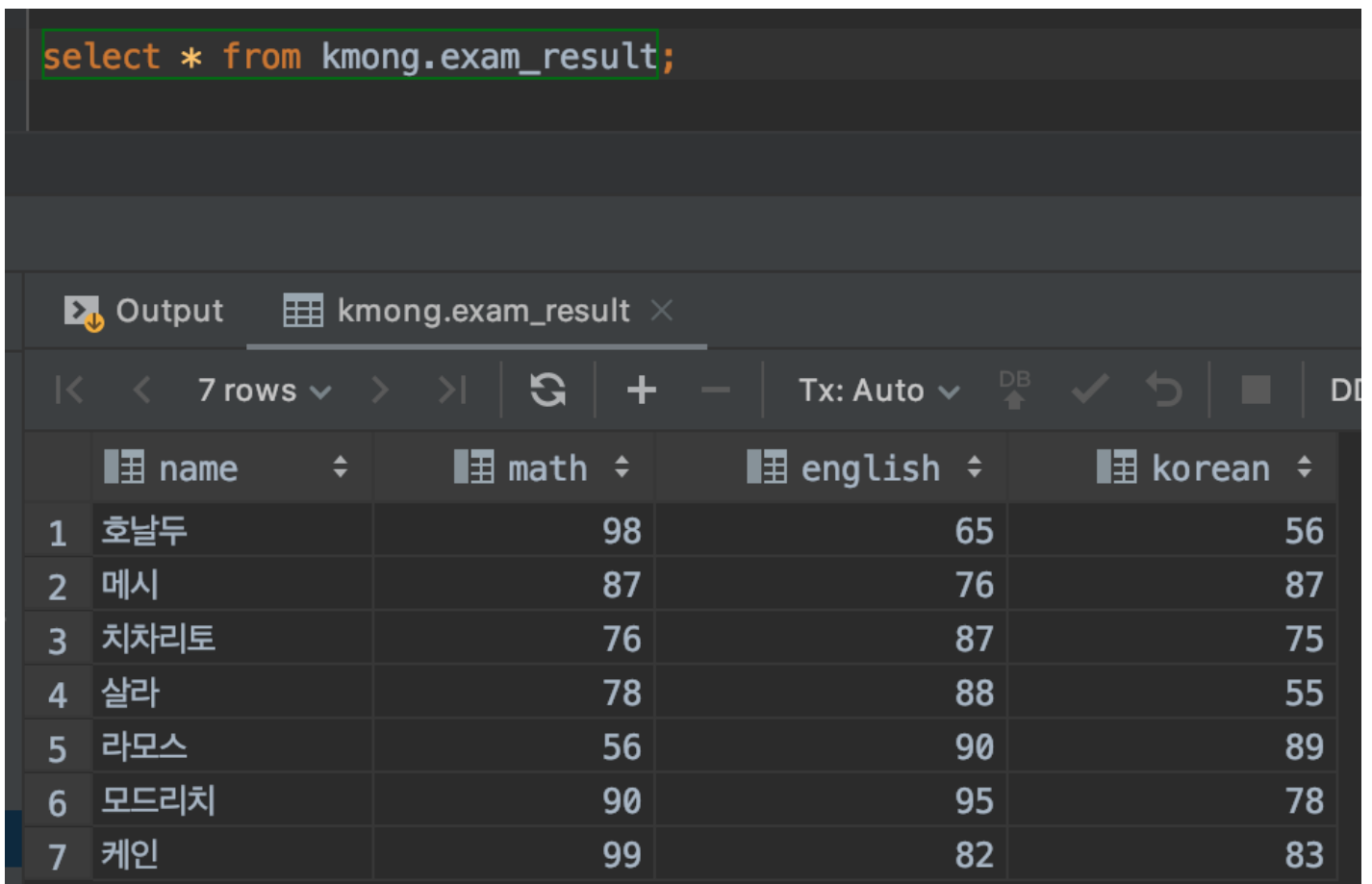
```
create table kmong.exam_result
(
    name    varchar(50),
    math    int(10),
    english int(10),
    korean  int(10)
) character set utf8;

INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('호날두', 98, 65, 56);
INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('메시', 87, 76, 87);
INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('치차리토', 76, 87, 75);
INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('살라', 78, 88, 55);
INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('라모스', 56, 90, 89);
INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('모드리치', 90, 95, 78);
INSERT INTO kmong.exam_result (name, math, english, korean) VALUES ('케인', 99, 82, 83);
```

이젠 데이터가 준비되었으면 산술 연산자 실습을 해보겠습니다.

먼저 데이터가 어떻게 들어갔는지 눈으로 확인을 해야겠죠.

```
select * from kmong.exam_result;
```



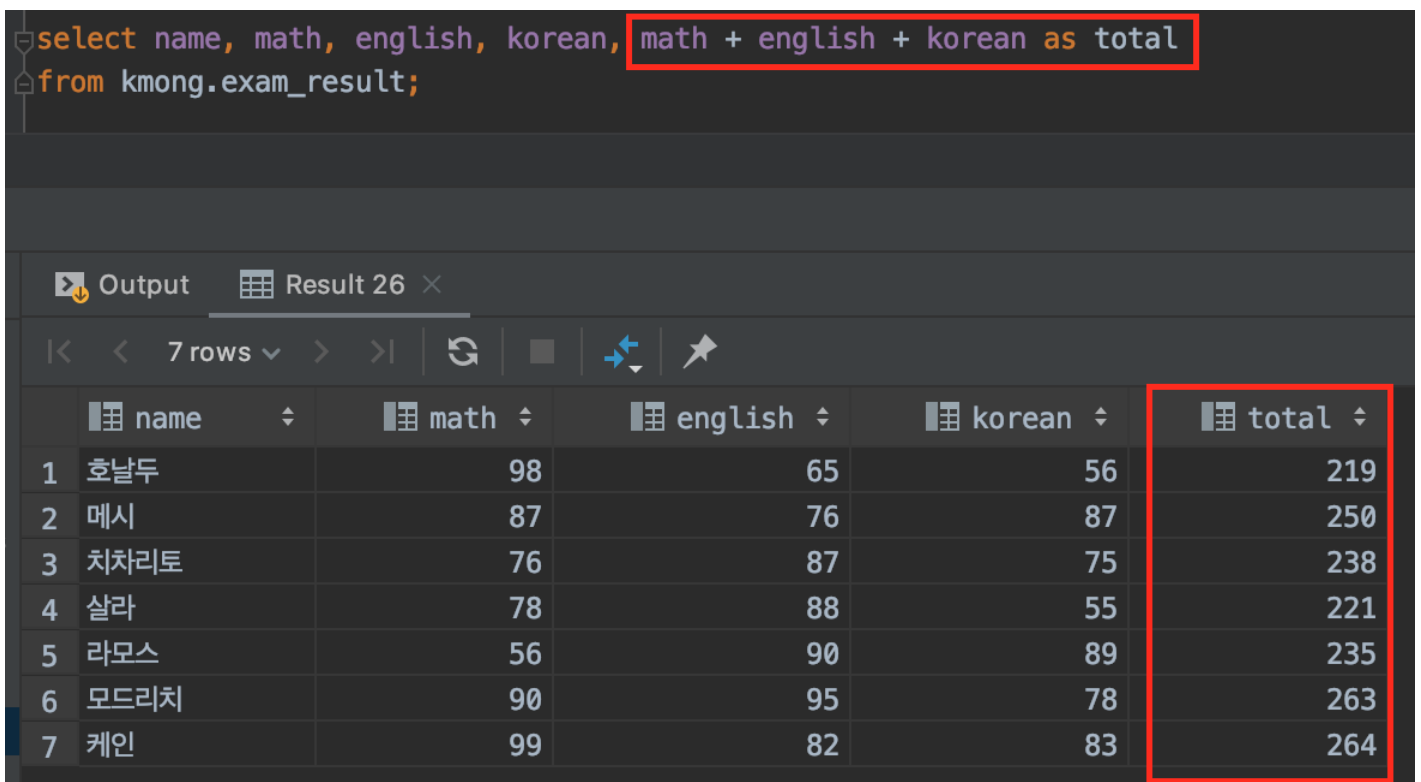
The screenshot shows a SQL IDE interface. At the top, a query editor contains the command `select * from kmong.exam_result;`, which is highlighted with a green box. Below the editor, the 'Output' tab is active, displaying the results of the query. The results are shown in a table with 4 columns: 'name', 'math', 'english', and 'korean'. There are 7 rows of data, numbered 1 through 7. The table data is as follows:

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 호날두 | 98 | 65 | 56 |
| 2 | 메시 | 87 | 76 | 87 |
| 3 | 치차리토 | 76 | 87 | 75 |
| 4 | 살라 | 78 | 88 | 55 |
| 5 | 라모스 | 56 | 90 | 89 |
| 6 | 모드리치 | 90 | 95 | 78 |
| 7 | 케인 | 99 | 82 | 83 |

칼럼을 우선 확인해보면, **name**이 있습니다. 그 옆으로 **math, english, korean**이라는 과목명이 칼럼명으로 되어 있습니다. 아래에는 사람 이름과 해당 과목들의 점수가 있습니다.

위 데이터에는 각 과목별 점수만 있지, 합계나 평균 같은 건 보이지 않습니다. 먼저 학생별 세 과목 합계 점수를 산술 연산자를 사용해서 구해보겠습니다.

```
select name, math, english, korean, math + english + korean as total
from kmong.exam_result;
```



```
select name, math, english, korean, math + english + korean as total
from kmong.exam_result;
```

| | name | math | english | korean | total |
|---|------|------|---------|--------|-------|
| 1 | 호날두 | 98 | 65 | 56 | 219 |
| 2 | 메시 | 87 | 76 | 87 | 250 |
| 3 | 치차리토 | 76 | 87 | 75 | 238 |
| 4 | 살라 | 78 | 88 | 55 | 221 |
| 5 | 라모스 | 56 | 90 | 89 | 235 |
| 6 | 모드리치 | 90 | 95 | 78 | 263 |
| 7 | 케인 | 99 | 82 | 83 | 264 |

위 그림을 보면 **math + english + korean as total**이라는 SQL 구문이 보이실 겁니다. 앞에 산술 연산자 외에 **as total** 이란 부분은 앞 시간에 공부했던 별칭(alias)입니다. 생각나시죠?

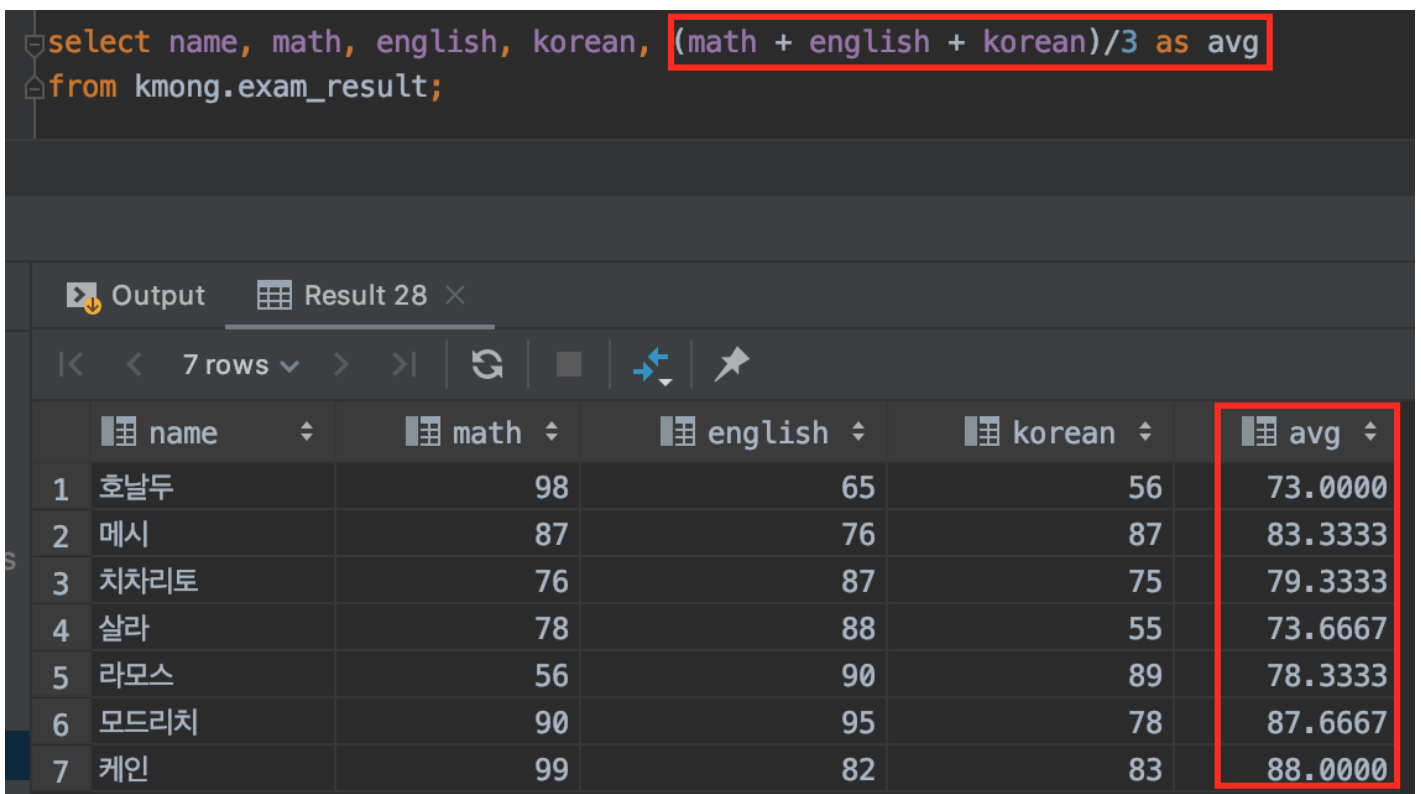
그리고 **math + english + korean** 부분을 보면 산술 연산자인 **+** 를 사용해서 각 시험 점수를 더한다라는 명령을 한 겁니다. 다시 말하면 각 시험 점수를 **(math + english + korean)** 더해서 **total**이라는 별칭으로 SQL을 실행하라는 이야기입니다.

그러니 아래 결과를 보면 total이라는 칼럼명으로 각 시험 점수가 더해져서 출력돼 있는 것을 확인할 수 있습니다.

이번에는 평균값을 구해 보도록 하겠습니다.

평균값을 구하기 위해서는 아래와 같이 SQL을 실행하면 됩니다.

```
select name, math, english, korean, (math + english + korean)/3 as avg
from kmong.exam_result;
```



The screenshot shows a SQL IDE interface. At the top, the query is entered: `select name, math, english, korean, (math + english + korean)/3 as avg from kmong.exam_result;`. The `(math + english + korean)/3 as avg` part is highlighted with a red box. Below the query editor, the 'Output' tab is selected, showing 'Result 28'. The result is a table with 7 rows. The columns are: name, math, english, korean, and avg. The 'avg' column is highlighted with a red box. The data is as follows:

| | name | math | english | korean | avg |
|---|------|------|---------|--------|---------|
| 1 | 호날두 | 98 | 65 | 56 | 73.0000 |
| 2 | 메시 | 87 | 76 | 87 | 83.3333 |
| 3 | 치차리토 | 76 | 87 | 75 | 79.3333 |
| 4 | 살라 | 78 | 88 | 55 | 73.6667 |
| 5 | 라모스 | 56 | 90 | 89 | 78.3333 |
| 6 | 모드리치 | 90 | 95 | 78 | 87.6667 |
| 7 | 케인 | 99 | 82 | 83 | 88.0000 |

아까 합계를 구한 것과 비슷해 보이지만 세 과목의 더한 값을 ()로 둘러싸고 3으로 나누어 줬습니다. 평균은 다들 구하실 수 있으시죠? ㅎㅎ 그리고 avg라고 별칭을 지정해주고 SQL을 실행하면 아래와 같이 avg라는 이름으로 컬럼이 생기고 평균값이 출력되고 있습니다.

간단하죠?

여러분들도 이젠 저 데이터를 이용해서 표준편차도 구해보시고, 여러 가지 실습을 직접 해보시기 바랍니다. 어렵지 않습니다.

2. where절에 비교 연산자를 사용해 보기

앞선 시간에 where절에 문자열을 넣어서 데이터중 조건에 맞는 일부 row만 출력하는 것을 배워보았습니다. 이번에는 **where** 절에 비교 연산자를 넣어서 출력하는 방법을 알아보겠습니다. 비교 연산자는 **<, >, =** 이런 것들을 이야기합니다.

위에서 실습했던 시험 점수 데이터를 계속 사용하겠습니다. 우선 **국어 점수가 80점 이상인 사람은 누구인지** 출력해보겠습니다.

```
select name, math, english, korean
from kmong.exam_result
where korean >= 80;
```

```
select name, math, english, korean
from kmong.exam_result
where korean >= 80;
```

Output kmong.exam_result

3 rows

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 메시 | 87 | 76 | 87 |
| 2 | 라모스 | 56 | 90 | 89 |
| 3 | 케인 | 99 | 82 | 83 |

where절에 **korean >= 80**이라는 조건을 주니 아래와 같이 korean이 80점 이상인 메시와 라모스, 케인이 출력되었습니다.

그럼 이번에는 바로 앞에서 배웠던 산술 연산자를 where절에 조건으로 넣고 비교 연산자를 이용해서 데이터를 걸러내는 것을 해보겠습니다. 세 과목의 평균점수가 80점 이상인 사람의 이름과 각 시험 점수를 출력하는 SQL입니다.

```
select name, math, english, korean
from kmong.exam_result
where (math+english+korean)/3 >= 80;
```

```
select name, math, english, korean
from kmong.exam_result
where (math+english+korean)/3 >= 80;
```

Output kmong.exam_result ×

3 rows ▾

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 메시 | 87 | 76 | 87 |
| 2 | 모드리치 | 90 | 95 | 78 |
| 3 | 케인 | 99 | 82 | 83 |

이렇게 산술 연산자를 select 절이 아닌 where 절에서도 사용이 가능합니다. **비교 연산자**와 함께 말이죠!

번외로 SQL의 where절에서 사용할 수 있는 다양한 연산자에 대해서 알아보겠습니다.

| 비교연산자 종류 | 설명 |
|----------|---------------|
| = | 같은 조건을 검색 |
| !=, <> | 같지 않은 조건을 검색 |
| > | 큰 조건을 검색 |
| >= | 크거나 같은 조건을 검색 |

| | |
|-----------------------|----------------------------|
| < | 작은 조건을 검색 |
| <= | 작거나 같은 조건을 검색 |
| BETWEEN a AND b | a 와 b 사이에 있는 값을 검색 |
| IN(a,b,c) | a,b,c 중 어느 하나 인 것을 검색 |
| like | 특정 패턴을 가지고 있는 조건을 검색 |
| is Null / is Not Null | NULL 인 값이나 NULL이 아닌 값을 검색 |
| a AND b | a, b 두 조건 모두를 만족하는 값을 검색 |
| a OR b | a 나 b 중 하나의 조건을 만족하는 값을 검색 |
| NOT a | a 가 아닌 모든 값을 검색 |

위에 있는 다양한 비교 연산자들을 이용해서 실습을 진행하시길 바랍니다.

3. order by 절을 사용 하여 정렬하여 출력 하기

데이터의 양이 많을 때는 데이터를 어떤 기준으로 정렬하여 보는 것이 편할 때가 있습니다. 이럴 때 SQL에서는 order by 절을 사용 하게 됩니다. order by 역시 select, from, where 등과 마찬가지로 키워드로 분류가 됩니다. 기본적으로 order by를 사용하게 되면 오름차순으로 정렬이 되며, 내림차순으로 정렬을 하고 싶을 때는 desc라는 옵션을 사용하게 됩니다.

문자의 경우 가, 나, 다 혹은 a, b, c 순으로 정렬이 되며, 날짜는 최근 날짜가 더 큰 값으로 인식되어 정렬됩니다.

다음 SQL문을 실행해 보도록 하겠습니다.

```
select *
from kmong.exam_result
order by math;
```

```
select *
from kmong.exam_result
order by math;
```

Output kmong.exam_result ×

7 rows ▾

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 라모스 | 56 | 90 | 89 |
| 2 | 치차리토 | 76 | 87 | 75 |
| 3 | 살라 | 78 | 88 | 55 |
| 4 | 메시 | 87 | 76 | 87 |
| 5 | 모드리치 | 90 | 95 | 78 |
| 6 | 호날두 | 98 | 65 | 56 |
| 7 | 케인 | 99 | 82 | 83 |

SQL의 맨 마지막에 order by math라고 입력을 하니 아래와 같이 수학 점수를 기준으로 오름차순 정렬이 된 것을 확인할 수 있습니다. 그럼 이번에는 desc 옵션을 주도록 하겠습니다.

```
select *
from kmong.exam_result
order by math desc;
```

```
select *
from kmong.exam_result
order by math desc;
```

Output kmong.exam_result ×

7 rows

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 케인 | 99 | 82 | 83 |
| 2 | 호날두 | 98 | 65 | 56 |
| 3 | 모드리치 | 90 | 95 | 78 |
| 4 | 메시 | 87 | 76 | 87 |
| 5 | 살라 | 78 | 88 | 55 |
| 6 | 치차리토 | 76 | 87 | 75 |
| 7 | 라모스 | 56 | 90 | 89 |

SQL의 마지막에 `order by math desc` 옵션을 주니 수학 점수가 높은 행부터 정렬된 것을 확인할 수 있습니다.

`order by` 에도 역시 산술 연산자를 사용해서 정렬하여 출력하게 할 수 있습니다.

```
select *
from kmong.exam_result
order by (math+english+korean)/3 desc;
```

```
select *
from kmong.exam_result
order by (math+english+korean)/3 desc;
```

| Output kmong.exam_result | | | | |
|--------------------------|------|------|---------|--------|
| 7 rows | | | | |
| | name | math | english | korean |
| 1 | 케인 | 99 | 82 | 83 |
| 2 | 모드리치 | 90 | 95 | 78 |
| 3 | 메시 | 87 | 76 | 87 |
| 4 | 치차리토 | 76 | 87 | 75 |
| 5 | 라모스 | 56 | 90 | 89 |
| 6 | 살라 | 78 | 88 | 55 |
| 7 | 호날두 | 98 | 65 | 56 |

위 그림에서와 같이 출력 값에서는 보이지 않지만 **order by (math+english+korean)/3 desc**라는 명령을 통해서 평균값이 높은 학생부터 정렬하여 출력이 된 모습을 확인할 수 있습니다.

그럼 아래 명령어는 무엇일까요?

```
select *
from kmong.exam_result
order by 3 ;
```

```
select *
from kmong.exam_result
order by 3 ;
```

Output kmong.exam_result

7 rows

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 호날두 | 98 | 65 | 56 |
| 2 | 메시 | 87 | 76 | 87 |
| 3 | 케인 | 99 | 82 | 83 |
| 4 | 치차리토 | 76 | 87 | 75 |
| 5 | 살라 | 78 | 88 | 55 |
| 6 | 라모스 | 56 | 90 | 89 |
| 7 | 모드리치 | 90 | 95 | 78 |

order by 3이라고 SQL을 작성해서 실행했습니다. order by 뒤에는 칼럼명이나 어떤 연산식을 넣을 수도 있지만 이렇게 숫자만 넣을 수도 있습니다.

order by 3의 의미는 SQL이 실행한 결과의 세 번째 컬럼을 기준으로 정렬 하라는 뜻 입니다. 그림을 보면 세번째 칼럼인 english 칼럼의 값을 기준으로 오름차순 정렬되어 출력된 결과를 확인하실 수 있습니다.

하지만, 이런 방법은 권장하는 방법은 아닙니다. 왜냐하면 복잡한 SQL에서 가독성이 떨어질 뿐 아니라, 정확하게 칼럼명을 명시하는 것이 나중에 SQL이나 테이블이 수정되어도 변함없는 결과를 가지고 올 수 있기 때문이죠. 하지만 이런 방법도 있는 걸 알아두실 필요가 있습니다.

4. 집합 연산자 사용하기

집합 연산자 사용하기 실습에 앞서 테이블 하나를 더 만들고, 데이터를 입력하겠습니다.

```
create table kmong.exam_result_2
(
```

```

name    varchar(50),
math    int(10),
english int(10),
korean  int(10)
) character set utf8;

```

```

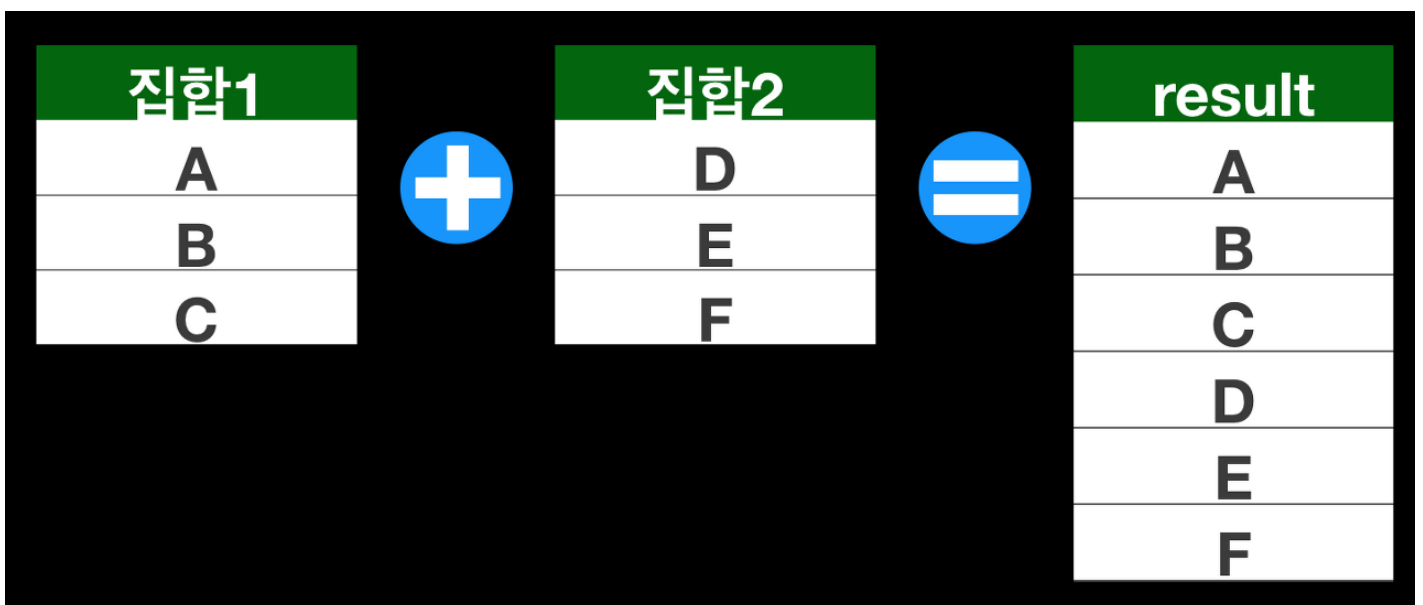
INSERT INTO kmong.exam_result_2 (name, math, english, korean) VALUES ('차범근', 78, 90, 78);
INSERT INTO kmong.exam_result_2 (name, math, english, korean) VALUES ('서정원', 68, 99, 68);
INSERT INTO kmong.exam_result_2 (name, math, english, korean) VALUES ('고종수', 84, 96, 98);
INSERT INTO kmong.exam_result_2 (name, math, english, korean) VALUES ('박지성', 67, 68, 75);
INSERT INTO kmong.exam_result_2 (name, math, english, korean) VALUES ('최순호', 88, 93, 68);

```

우선 집합 연산자의 종류에 대한 내용을 확인하고 넘어가겠습니다.

그리고, 집합이라고 하면 데이터베이스에서는 기본적으로 한 테이블을 하나의 집합이라고 하고, SQL에서 하나의 select 문으로 나오는 **데이터셋을 집합**이라고도 표현합니다. 집합 연산자에 대한 내용 확인해보겠습니다.

| 집합연산자 종류 | 내용 |
|-----------|---|
| UNION | 두 집합을 더해서 결과를 출력한다. 중복 값 제거하고 정렬을 수행한다. |
| UNION ALL | 두 집합을 더해서 결과를 출력한다. 중복 제거와 정렬을 하지 않는다. |
| INTERSECT | 두 집합의 교집합 결과를 정렬하여 출력한다. |
| MINUS | 두 집합의 차집합 결과를 정렬하고 출력한다. SQL의 순서가 중요하다. |



우선 **UNION**으로 SQL을 실행하는 예제를 보겠습니다.

```

select *
from kmong.exam_result
union

```

```
select *  
from kmong.exam_result_2
```

```
select *  
from kmong.exam_result  
union  
select *  
from kmong.exam_result_2
```

Output kmong.exam_result

12 rows

| | name | math | english | korean |
|----|------|------|---------|--------|
| 1 | 호날두 | 98 | 65 | 56 |
| 2 | 메시 | 87 | 76 | 87 |
| 3 | 치차리토 | 76 | 87 | 75 |
| 4 | 살라 | 78 | 88 | 55 |
| 5 | 라모스 | 56 | 90 | 89 |
| 6 | 모드리치 | 90 | 95 | 78 |
| 7 | 케인 | 99 | 82 | 83 |
| 8 | 차범근 | 78 | 90 | 78 |
| 9 | 서정원 | 68 | 99 | 68 |
| 10 | 고종수 | 84 | 96 | 98 |
| 11 | 박지성 | 67 | 68 | 75 |
| 12 | 최순호 | 88 | 93 | 68 |

이렇게 UNION을 이용해서 두 select 결과 (집합)을 합칠 수 있습니다.

다음은 **UNION**과 **UNION ALL**의 차이점도 한번 확인해보겠습니다.

```
select *
from kmong.exam_result
union
select *
from kmong.exam_result;
```

```
select *
from kmong.exam_result
union all
select *
from kmong.exam_result;
```

The left screenshot shows the result of a UNION query. The output table has 7 rows, which are the unique records from the kmong.exam_result table. The right screenshot shows the result of a UNION ALL query. The output table has 14 rows, which are all records from the kmong.exam_result table, including duplicates.

| | name | math | english | korean |
|---|------|------|---------|--------|
| 1 | 호날두 | 98 | 65 | 56 |
| 2 | 메시 | 87 | 76 | 87 |
| 3 | 치차리토 | 76 | 87 | 75 |
| 4 | 살라 | 78 | 88 | 55 |
| 5 | 라모스 | 56 | 90 | 89 |
| 6 | 모드리치 | 90 | 95 | 78 |
| 7 | 케인 | 99 | 82 | 83 |

| | name | math | english | korean |
|----|------|------|---------|--------|
| 1 | 호날두 | 98 | 65 | 56 |
| 2 | 메시 | 87 | 76 | 87 |
| 3 | 치차리토 | 76 | 87 | 75 |
| 4 | 살라 | 78 | 88 | 55 |
| 5 | 라모스 | 56 | 90 | 89 |
| 6 | 모드리치 | 90 | 95 | 78 |
| 7 | 케인 | 99 | 82 | 83 |
| 8 | 호날두 | 98 | 65 | 56 |
| 9 | 메시 | 87 | 76 | 87 |
| 10 | 치차리토 | 76 | 87 | 75 |
| 11 | 살라 | 78 | 88 | 55 |
| 12 | 라모스 | 56 | 90 | 89 |
| 13 | 모드리치 | 90 | 95 | 78 |
| 14 | 케인 | 99 | 82 | 83 |

같은 테이블을 **UNION** , **UNION ALL**로 각각 합쳐봤습니다. 중복을 제거하는 **UNION**에서와 제거하지 않는 **UNION ALL**의 차이점이 한눈에 보입니다.

이번엔 **INTERSECT**와 **MINUS**의 차이점을 비교해보겠습니다. 지금 수업을 진행 중인 **mysql**의 **5.7.29-0 ubuntu0.18.04.1** 버전에서는 **INTERSECT**와 **MINUS** 기능이 지원하지 않습니다. 다른 방식으로 해당 기능을 구현해야 하는데, 일단 직관적인 이해를 위해서 **oracle 11g** 버전에서 해당 기능의 실습을 진행했습니다.

실습을 진행하기 위해서 두 과목의 수강생 테이블을 만들고 데이터를 넣겠습니다.

```
create table math_student
(
    name varchar(50),
```



```

student_no varchar(10)
);

create table korean_student
(
    name varchar(50),
    student_no varchar(10)
);

INSERT INTO math_student (name, student_no) VALUES ('조단', '111');
INSERT INTO math_student (name, student_no) VALUES ('호나우두', '112');
INSERT INTO math_student (name, student_no) VALUES ('나달', '113');
INSERT INTO math_student (name, student_no) VALUES ('조코비치', '114');

INSERT INTO korean_student (name, student_no) VALUES ('조단', '111');
INSERT INTO korean_student (name, student_no) VALUES ('호나우두', '112');
INSERT INTO korean_student (name, student_no) VALUES ('조현우', '201');
INSERT INTO korean_student (name, student_no) VALUES ('루이스', '202');

```

각각의 테이블을 select 해서 확인해보겠습니다.

The image shows two side-by-side screenshots of a SQL IDE. The left screenshot shows the query `select * from kmong.math_student;` and its result set with 4 rows: 조단 (111), 호나우두 (112), 나달 (113), and 조코비치 (114). The right screenshot shows the query `select * from kmong.korean_student;` and its result set with 4 rows: 조단 (111), 호나우두 (112), 조현우 (201), and 루이스 (202). The 'korean_student' query is highlighted with a green box in the original image.

| name | student_no |
|--------|------------|
| 1 조단 | 111 |
| 2 호나우두 | 112 |
| 3 나달 | 113 |
| 4 조코비치 | 114 |

| name | student_no |
|--------|------------|
| 1 조단 | 111 |
| 2 호나우두 | 112 |
| 3 조현우 | 201 |
| 4 루이스 | 202 |

이젠, 이 두 테이블의 데이터를 이용해서 INTERSECT와 MINUS 연산자 사용 시 결과 비교를 해보도록 하겠습니다.

SQL은 아래와 같이 실행합니다.

```
select *
from math_student
intersect
select *
from korean_student;
```

```
select *
from math_student
minus
select *
from korean_student;
```

```
select *
from math_student
intersect
select *
from korean_student;
```

Output STRICKY.MATH_STUDENT

| | NAME | STUDENT_NO |
|---|------|------------|
| 1 | 조단 | 111 |
| 2 | 호나우두 | 112 |

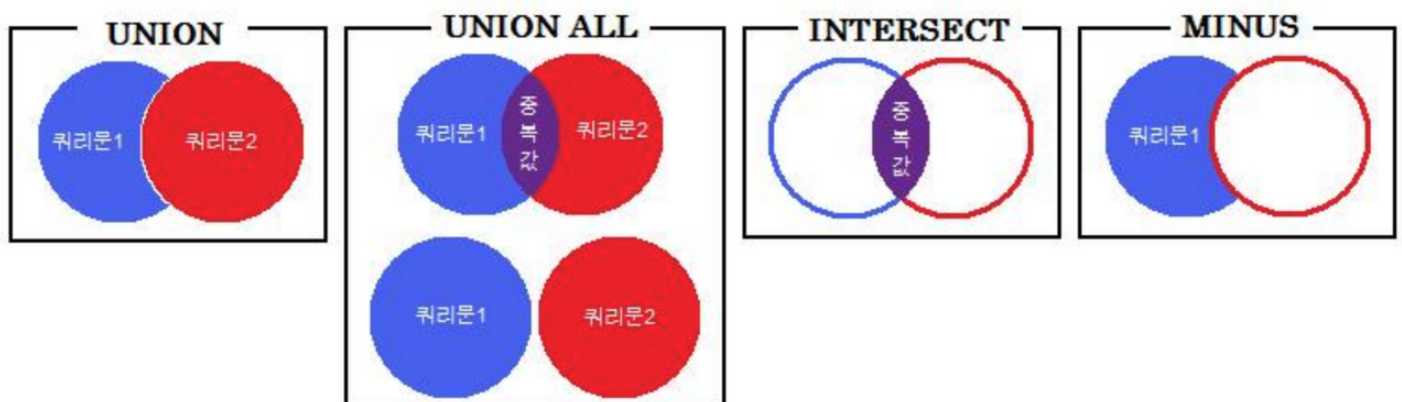
```
select *
from math_student
minus
select *
from korean_student;
```

Output STRICKY.MATH_STUDENT

| | NAME | STUDENT_NO |
|---|------|------------|
| 1 | 나달 | 113 |
| 2 | 조코비치 | 114 |

위에서 select 한 내용과 아래에서 INTERSECT, MINUS 연산자를 사용한 결과와 비교를 해보시면 나오는 결과에 대해서 충분히 이해가 가능할 것입니다.

지금까지 확인한 집합 연산자에 대해서 그림으로 다시 상기해 이해해보겠습니다.



집합 연산자를 잘 사용하면 복잡한 SQL을 간결하게 표현할 수 있습니다.

SQL 개인강의 안내

open.kakao.com/o/szfhqYec

| | |
|--|---|
| | Database/남/db개발님의 오픈프로필 open.kakao.com |
|--|---|

자, 이렇게 select를 잘 이용하는 방법에 대해서 공부해 봤습니다. 여기까지 잘 따라서 실습을 하셨으면 select에 대해서 어느 정도 감을 잡았을 거라고 생각합니다. 이젠 뼈대를 만들었으니, 이젠 앞으로의 내용은 살을 붙여가는 과정이라고 생각하시면 됩니다.

다음 시간은 단일행 함수 사용하기에 대해서 공부해 보겠습니다. 앞으로도 많이 봐주시고 따라서 공부해 주셨으면 좋겠습니다.

감사합니다.