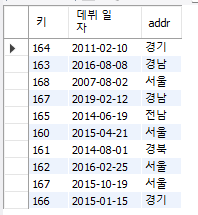
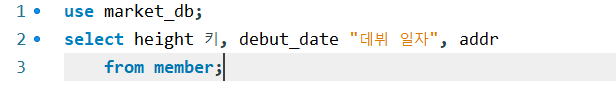
- 열의 이름이 헷갈릴때는 임의로 이름을 지정해서 볼수도 있다

- 임의로 이름을 지정할 때 띄어쓰기가 들어가는 경우 “”(큰따옴표)로 묶어준다.



# ※ OR = IN

SELECT mem\_name, addr

FROM member

WHERE addr = '경기' OR addr = '전남' OR addr = '경남';

=> SELECT mem\_name, addr

FROM member

WHERE addr IN('경기', '전남', '경남');

# ※ %는 여러 글자 \_는 한 글자

- SELECT \*

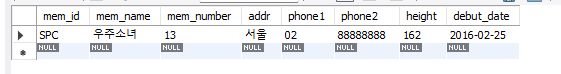
FROM member

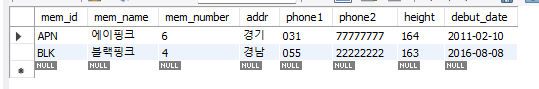
WHERE mem\_name LIKE '우%';

- SELECT \*

FROM member

WHERE mem\_name LIKE '\_\_핑크';





# ※ ORDER BY절

- 결과의 값이나 개수에 대해서는 영향을 미치지 않지만, 결과가 출력되는 순서를 조절

Ex) SELECT mem\_id, mem\_name, debut\_date

FROM member

ORDER BY debut\_date; (기본적으로 오름차순 ASC)

SELECT mem\_id, mem\_name, debut\_date

FROM member

ORDER BY debut\_date DESC; (DESC는 내림차순)

Ex2) WHERE절은 ORDER BY절 앞에 나와야 한다.

SELECT mem\_id, mem\_name, debut\_date, height

FROM member

ORDER BY height DESC

WHERE height >= 164; -- 오류 발생

수정 후 ->

SELECT mem\_id, mem\_name, debut\_date, height

FROM member

WHERE height >= 164

ORDER BY height DESC;

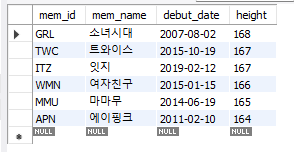
Ex3) 값이 같은 경우 조건식을 여러 개 쓸 수 있다.

SELECT mem\_id, mem\_name, debut\_date, height

FROM member

WHERE height >= 164

ORDER BY height DESC, debut\_date ASC;



-> 먼저 정렬한 조건식에 의해 키가 167로 같은 경우 데뷔일자를 기준으로 오름차순 정렬

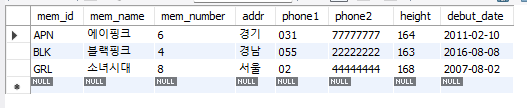
# ※ LIMIT

Ex) SELECT \*

FROM member

LIMIT 3;

-> 전체 중에 위에 있는 3개 행만 조회



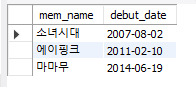
Ex2) SELECT mem\_name, debut\_date

FROM member

ORDER BY debut\_date

LIMIT 3;

-> 데뷔일자가 빠른 3개 행만 조회



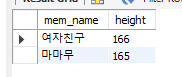
Ex3) SELECT mem\_name, height

FROM member

ORDER BY height DESC

LIMIT 3,2;

-> 3등부터 2개를 뽑아서 조회



# ※ DISTINCT

- 중복된 값을 빼고 보여줌

SELECT DISTINCT addr FROM member;

# ※ GROUP BY절

- 말 그대로 그룹으로 묶어주는 역할

- 집계 함수와 함께 사용됨

- SUM, AVG, MIN, MAX, COUNT, VOUNT(DISTINCT)

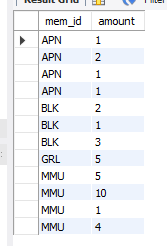
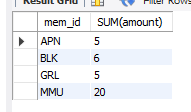
Ex) SELECT mem\_id, amount FROM buy ORDER BY mem\_id;

(mem\_id 별로 구매했던 양을 보여주지만 하나 하나 계산하기가 어려움 그래서 사용하게 되는게 GROUP BY절) ----------------->

SELECT mem\_id, SUM(amount) FROM buy GROUP BY mem\_id;

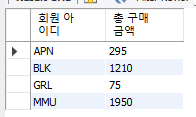
= SELECT mem\_id "회원 아이디", SUM(amount) "총 구매 개수"

FROM buy GROUP BY mem\_id; // 별칭을 줘서 보기 편리하게 볼 수도 있음

🡺  🡺 

Ex2) SELECT mem\_id "회원 아이디", SUM(price\*amount) "총 구매 금액"

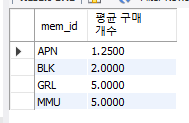
FROM buy GROUP BY mem\_id;



Ex3) SELECT mem\_id, AVG(amount) "평균 구매 개수"

FROM buy

GROUP BY mem\_id;



# ※ COUNT

Ex) SELECT COUNT(\*) FROM member;

-> 모든 행의 개수를 조회

Ex2) SELECT COUNT(phone1) "연락처가 있는 회원" FROM member;

-> Null 값은 빼고 개수를 조회해준다.

# ※ HAVING

- 조건절에 집계함수가 와야함

Ex) SELECT mem\_id "회원 아이디", SUM(price\*amount) "총 구매 금액"

FROM buy

WHERE SUM(price\*amount) > 1000

GROUP BY mem\_id; // 오류 발생(집계함수의 조건에는 WHERE절을 사용하면 안됨)

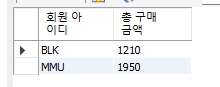
--------------> 수정 후

SELECT mem\_id "회원 아이디", SUM(price\*amount) "총 구매 금액"

FROM buy

GROUP BY mem\_id

HAVING SUM(price\*amount) > 1000 ;



Ex-1) 위의 결과문에서 내림차순으로 정렬한 SQL문

SELECT mem\_id "회원 아이디", SUM(price\*amount) "총 구매 금액"

FROM buy

GROUP BY mem\_id

HAVING SUM(price\*amount) > 1000

ORDER BY SUM(price\*amount) DESC;