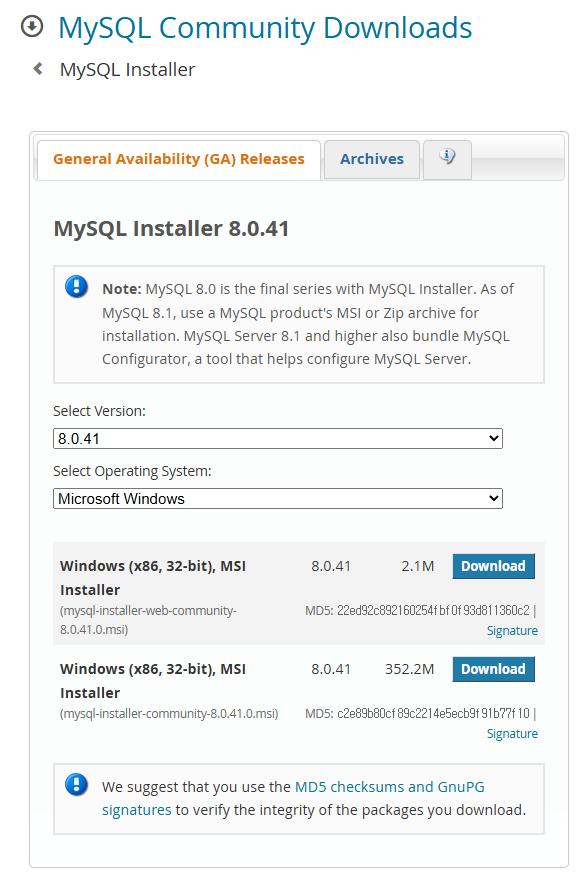
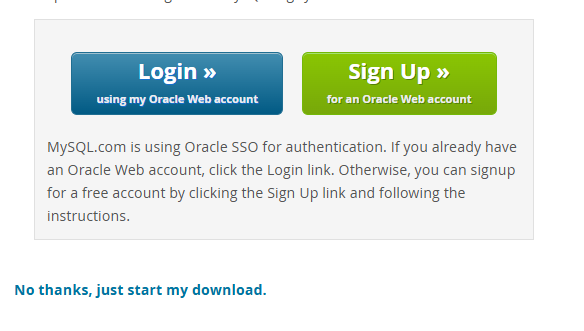
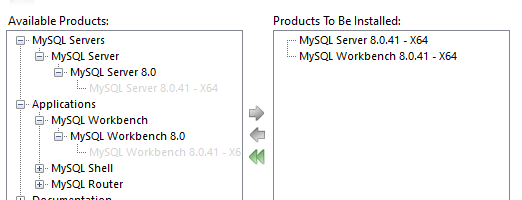
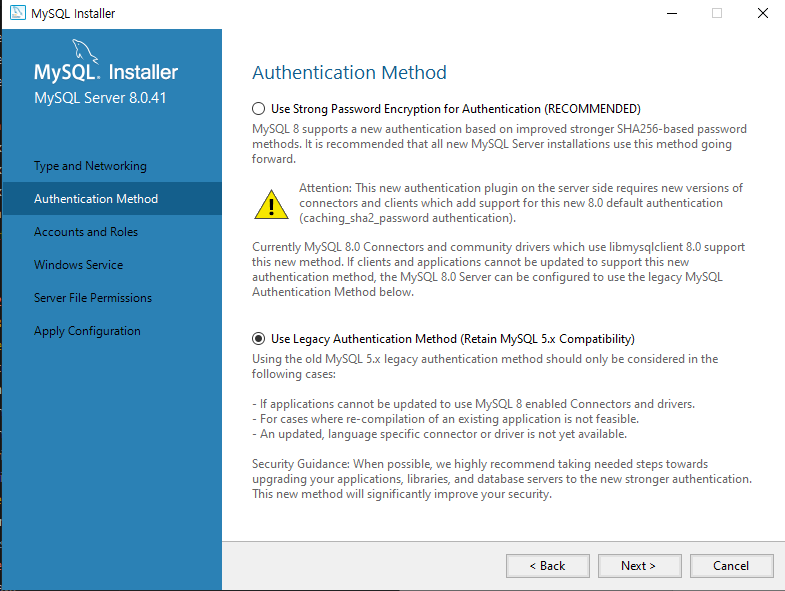
**MySQL 설치**

1. 구글 – mysql 검색
2. MySQL Installer8.0.41









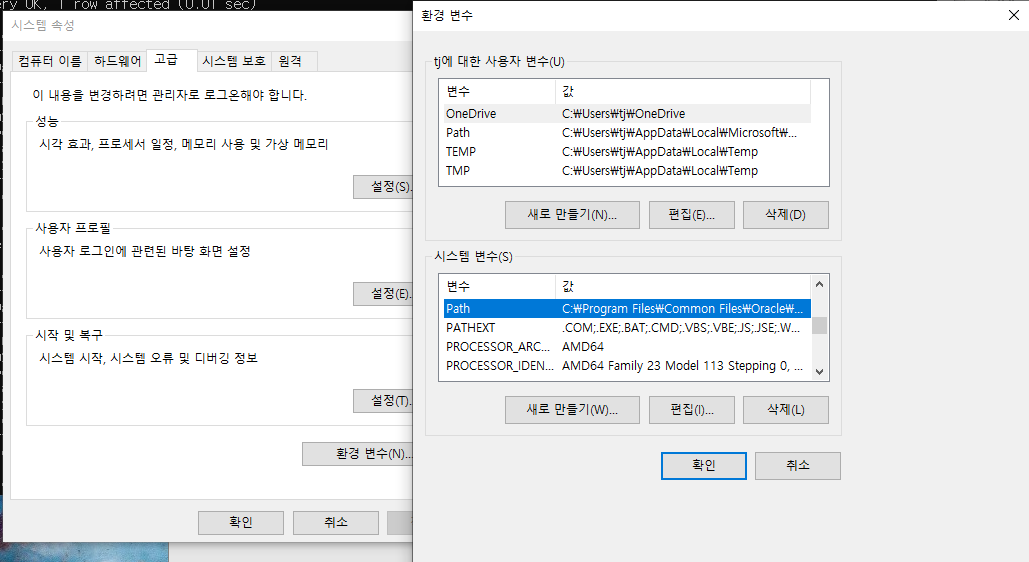
Password : 1234

나머지는 전부 next

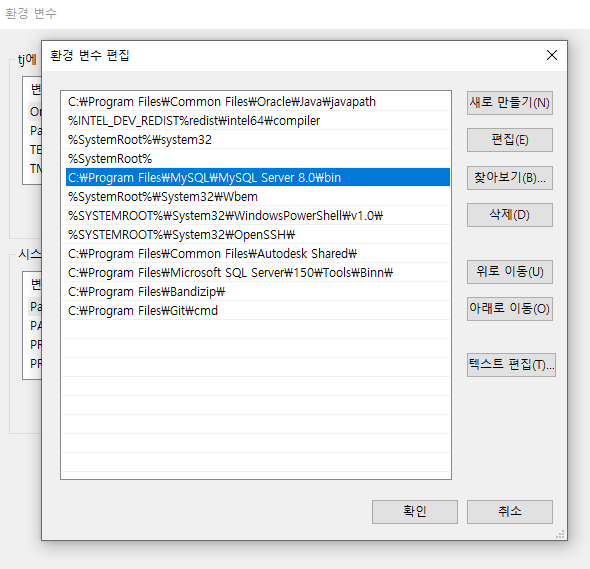
* **환경변수 설정**

검색 – 시스템 환경 변수 편집 – 환경변수

Path – 편집



새로 만들기 – 찾아보기 – MySQL의 bin 추가



**cmd에서 MySQL 접속**

mysql -u root -p

password 1234 입력

* **대소문자 구분 없음**

**데이터베이스 목록 보기 :** show databases;

**데이터베이스 생성 :** create database test;

**데이터 베이스 삭제 :** drop database test; -> (drop database ‘db명’)

**데이터 베이스 사용(선택) :** use sample; -> (use ‘db명’)

**use 샘플 데이터베이스 안의 테이블 보기 :** show tables;

**현재 로그인한 사용자 :** select current\_user(); -> 결과 : root@localhost

**유저 생성 :** create user eunji@localhost identified by '1234'; **->** 비밀번호 1234

**권한 부여 :** GRANT ALL ON test.\* TO eunji@localhost; -> Test DB의 모든 권한을 줌

* \*.\* : 모든 db의 모든 권한.
* Eunji@% : 은지한테 모든 권한 줌
* GRANT **SELECT, INSERT** ON **\*.\*** TO **eunji**@localhost : select, insert 권한만 줌
* GRANT **SELECT** ON **test.\*** TO eunji@localhost : test db의 select 권한만 줌
* **Test.friend** : 테스트 DB의 friend 테이블만 권한 줌

**메모리에 적용 :** flush privileges; -> 이 설정이 저장됨

**권한 삭제 :** REVOKE INSERT, UPDATE ON test.\* FROM eunji; -> insert,update 권한 삭제

* **사용자 전체 권한 삭제 :** REVOKE ALL ON \*.\* FROM eunji;
* **사용자 전체 쓰기 권한 삭제 :** REVOKE INSERT ON \*.\* TO eunji@localhost;

**유저 삭제 :** DROP USER eunji@localhost;

**Localhost와 %의 차이**

* **Localhost :** 자신의 컴퓨터에서만 접속 허용
* **% :** 어디서든 접속 가능 (보안 위험, 방화벽 필수)

Create user [eunji@192.168.1.10](mailto:eunji@192.168.1.10) identified by ‘1234’;

**회원 테이블 만들기**

CREATE TABLE member

( mem\_id CHAR(8) NOT NULL PRIMARY KEY, # primary key로 설정

mem\_name VARCHAR(10) NOT NULL,

mem\_number TINYINT NOT NULL,

addr CHAR(2) NOT NULL,

phone1 CHAR(3) NULL,

phone2 CHAR(8) NULL,

height TINYINT UNSIGNED NULL,

debut\_date DATE NULL

PRIMARY KEY(num); -> 따로 유일키 지정해도 됨

);

create table users(

-> id int auto\_increment primary key,

-> name varchar(50) not null,

-> email varchar(100) unique,

-> created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP );

-> varchar() : 메모리 자동 조절

Create\_at : 시간 자동 기록

Tinyint : 자바에 byte와 비슷(-128~127)

Tinyint unsigned : 음수 불가. 양수 범위가 두 배로 커짐. (0~255)

Primary key : 유일키

INSERT INTO member(name, tel, address) VALUES ( , , ); -> 선택해서 삽입

INSERT INTO member VALUES( , , );

* 앞에 auto\_increment가 있을 경우, 그 자리에 null 추가해줘야 함(다른 값 넣으면 그 값이 들어감)

DESC member; -> 테이블 구조 보여줌

SELECT \* FROM member; -> 모든 내용 보여줌

SELECT mem\_id, mem\_name FROM member; -> member 테이블에서 mem\_id, mem\_name 만 보여줌

RENAME TABLE bye TO buy; -> 테이블 이름을 bye->buy로 변경

**필드 추가**

ALTER TABLE member **ADD** post\_num CHAR(20); -> 맨 뒤에 필드 추가

ALTER TABLE member ADD post\_num CHAR(20) AFTER mem\_number; -> mem\_nember 뒤에 필드 추가

ALTER TABLE member DROP post\_num; -> post\_num 필드 삭제

ALTER TABLE member CHANGE phone1 tel1 char(4); -> phone1을 tel1으로 바꿈

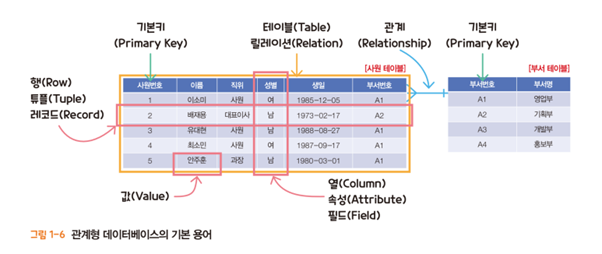
**필드 추가(범용적)**

ALTER TABLE person **ADD COLUMN** agee double; -> column 추가

ALTER TABLE person CHANGE COLUMN agee age double; -> column 명 변경

ALTER TABLE person MODIFY COLUMN age int; -> 데이터 타입을 int로 변경

ALTER TABLE person DROP COLUMN age; -> age column 삭제



**조건식 조회**

SELECT id, name, address, gender FROM member WHERE gender='w';

SELECT name, tel FROM member WHERE name LIKE ‘박%’;

* ‘박’으로 시작하는 모든 이름
* Not like – 포함하지 않는

**레코드 정렬**

SELECT 필드명1, 필드명2 FROM 테이블명 ORDER BY 필드명;

* 필드명 순으로 오름차순 정렬

SELECT 필드명1, 필드명2 FROM 테이블명 ORDER BY 필드명 DESC;

* 내림차순 정렬

**레코드 수정**

UPDATE 테이블명 SET 필드명=필드값 WHERE 조건식;

UPDATE member SET age=37 WHERE name=’정한나’;

**레코드 삭제**

DELETE FROM 테이블명 WHERE 조건식

DELETE FROM member WHERE name=’김수련’;

DELETE FROM member; -> 전체 레코드 삭제

C:\Users\tj>mysqldump -uroot -p1234 test > test.sql

* 저 위치에 test 데이터베이스가 test.sql 파일로 저장됨

DROP TABLE IF EXISTS `person`; -> person 테이블이 이미 있으면 지워라

SELECT 고객번호, 담당자명, 고객회사명, 마일리지 as 포인트, 마일리지\*1.1 as "10% 인상된 마일리지" from 고객;

* 마일리지\*1.1 가능. 마일리지를 "10% 인상된 마일리지"로 변경 가능.

select 고객번호, 담당자명, 도시, 마일리지 as 포인트 from 고객 where 도시='서울특별시' order by 마일리지 desc;

**Limit : 개수 제한**

Select \* from 고객 limit 3; ->고객테이블에서 1행부터 시작하여 3개의 고객정보를 조회

Select \* from 고객 limit 10, 3; -> 10행부터 3개 보여줌

Select \* from 고객 order by 마일리지 desc limit 3; -> 마일리지 많은 3명

Select **distinct** 도시 from 고객; -> 중복 데이터 제거

Select 23+5 as 더하기,

23-5 as 빼기,

5\*2 as 곱하기,

23/5 as 실수나누기,

23 div 5 as 정수나누기,

23%5 as 나머지1,

23 mod 5 as 나머지2;

Select 23>=5, 23<=5, 5>5 , 23<23, 23=23, 23!=5, 23<>23;

* 1, 0, 0, 0, 1, 1, 0

select \* from 고객 where 담당자직위 != '대표 이사';

select \* from 고객 where 담당자직위 <> '대표 이사';

select \* from 고객 where 지역=""; -> 빈문자열

select \* from 고객 where 지역 is null; -> null

**빈 문자열을 null로 변경할 건데 안전모드가 걸려 있는 경우(Error 1175)**

방법1)

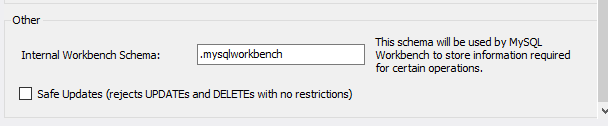
Set SQL\_SAFE\_UPDATES = 0;

Update 고객 set 지역=null where 지역=’’ and 고객번호 is not null;

* 빈 문자열인 지역을 null로 변경. 근데 update가 안전 모드로 설정되어 있어서 안 됨. 그래서 안전모드를 0으로 바꾸고 null로 변경.
* 끝나면 set SQL\_SAFE\_UPDATES = 1; 해서 원래대로 변경

방법2)

Edit> Preferences > SQL Editor > Safe Updates 옵션 체크 해제 후 프로그램 재실행



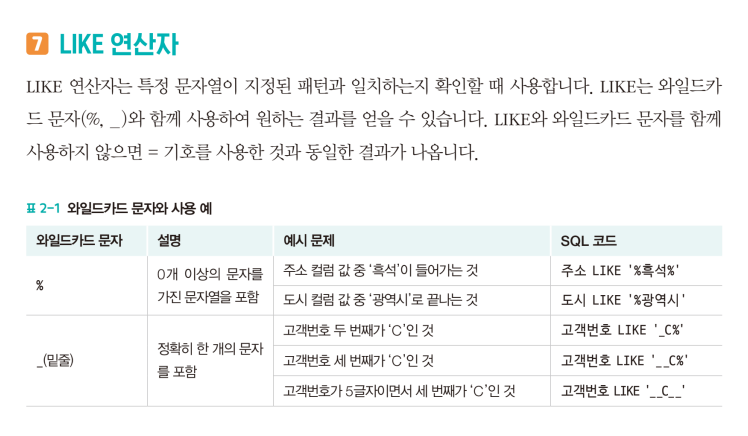
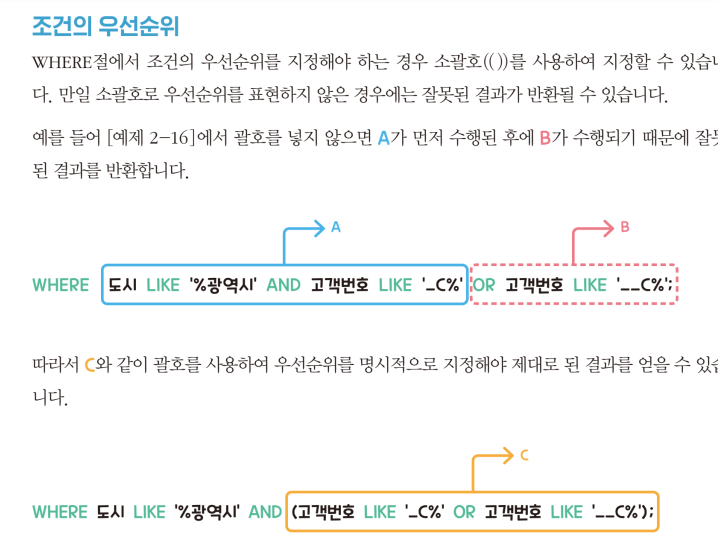
SELECT고객번호, 담당자명, 담당자직위 FROM 고객 WHERE 담당자직위 IN ('영업 과장','마케팅 과장'); -> or 조건이 여러 개일 경우 **IN** 사용

select 담당자명, 마일리지 from 고객 where 마일리지 BETWEEN 100000 AND 200000 ;

* 10만 ~ 20만

SELECT \* FROM 고객 WHERE 도시 LIKE '%광역시' AND (고객번호 like '\_C%' or 고객번호 like '\_\_C%'); -> 두 번째 글자가 C인 경우 or 세 번째 글자가 C인 경우

조건의 우선순위



글자의 길이

SELECT CHAR\_LENGTH('HELLO') // 영문자 개수 반환. 5

,LENGTH('HELLO') // 영문자 바이트 수. 5

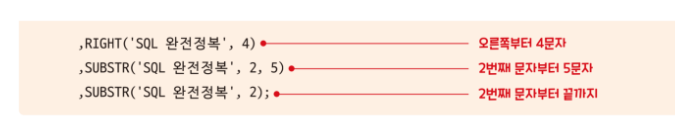
,CHAR\_LENGTH('안녕') // 문자 개수. 2

,LENGTH('안녕'); // 문자 바이트 수. 5

SELECT CONCAT(‘DREAMES’, ‘COME’, ‘TRUE’) // 붙어서 나옴. DREAMESCOMETRUE

, CONCAT\_WS(‘-‘, ‘2025’, ‘04’, ‘10’); // 2025-04-10

SELECT LEFT('SQL 완전정복', 3); // 왼쪽부터 3글자만 추출



**SUBSTR(문자열, 시작위치, 길이) = SUBSTRING(문자열, 시작위치, 길이)**

**(SUBSTRING이 표준 SQL)**

**SUBSTRING\_INDEX(문자열, 특정 구분자, 개수) – 개수 양수: 왼쪽부터 개수개까지 구분자를 기준으로 잘라서 리턴**

SELECT SUBSTRING\_INDEX('서울시/동작구/흑석로', '/', 2) // 서울시/동작구

,SUBSTRING\_INDEX('서울시/동작구/흑석로', '/', -2); // 동작구/흑석로

**LPAD(문자열, 길이, 앞 채울 문자열) OR RPAD(문자열, 길이, 뒤 채울 문자열)**

SELECT LPAD('SQL',10,'#') // 문자 포함 10개. 앞 공백은 #으로 채움

,RPAD('SQL',5,'\*'); // 문자 포함 5개. 뒤 공백은 \*로 채움

SELECT LTRIM(‘ SQL ‘), RTRIM(‘ SQL ‘), TRIM(‘ SQL ‘) // 왼쪽, 오른쪽, 양쪽 공백 제거

SELECT TRIM(BOTH 'abc' FROM 'abcSQLabcabc') // SQL. 모든 abc 제거

,TRIM(LEADING 'abc' FROM 'abcSQLabcabc') // SQLabcabc. 왼쪽

,TRIM(TRAILING 'abc' FROM 'abcSQLabcabc'); // abcSQL. 오른쪽

SELECT FIELD('JAVA', 'SQL', 'JAVA', 'C') // 앞의 JAVA를 뒤의 문자 중에서 찾기

,FIND\_IN\_SET('JAVA', 'SQL,JAVA,C') // 앞의 JAVA를 ,로 구분된 뒤에서 찾기

,INSTR('네 인생을 살아라', '인생') // 앞 문장 중 뒤의 단어 시작 위치 찾기

,LOCATE('인생', '네 인생을 살아라'); // 앞 문자를 뒤의 문장에서 시작위치 찾기

SELECT REPEAT('\*',5);// \* 5개 반복

SELECT REPLACE('010.1234.5678', '.', '-'); // 맨 앞 문자열의 . 을 – 로 바꾸기

SELECT CEILING(123.45) // 올림

, FLOOR(123.55) // 내림

,ROUND(123.55) // 반올림

, ROUND(123.56,1) // 소수 첫째 자리까지 반올림

,TRUNCATE(123.55,1); // 소수 첫째 자리까지 내림

SELECT MOD(203, 4), 203 % 4 , 203 MOD 4; // 나머지

SELECT POWER(2,3) // 거듭제곱 (2의 세제곱 = 8)

, SQRT(16) // 제곱근 (루트 16 = 4)

, RAND() // 랜덤값

, RAND(100)

, ROUND(RAND() \* 100) // 0~100

**날짜**

SELECT NOW(); // 현재 날짜 시간

SELECT SYSDATE(), // 현재 날짜 시간

CURDATE(), // 현재 날짜

CURTIME(); // 현재 시간

SELECT NOW()

,YEAR(NOW()) // 연도

,QUARTER(NOW()) // 분기

,MONTH(NOW())

,DAY(NOW())

,HOUR(NOW())

,MINUTE(NOW())

,SECOND(NOW());

SELECT NOW()

,DATEDIFF('2026-12-20',NOW()) // 앞 날짜 – 뒤 날짜

, DATEDIFF(NOW(),'2026-12-20')

, TIMESTAMPDIFF(YEAR, NOW(), '2026-12-20') // 뒤 날짜 – 앞 날짜

, TIMESTAMPDIFF(MONTH, NOW(), '2026-12-20') // 20개월 남음

, TIMESTAMPDIFF(DAY, NOW(), '2026-12-20');

SELECT NOW()

, ADDDATE(NOW(), 50) // 50일 후

, ADDDATE(NOW(), INTERVAL 50 DAY) // 50일 후

, ADDDATE(NOW(), INTERVAL 50 MONTH)

, SUBDATE(NOW(), INTERVAL 50 HOUR); // 50시간 전

SELECT NOW()

, LAST\_DAY(NOW()) // 이번 달의 마지막 날짜

, DAYOFYEAR(NOW()) // 1년 중 며칠 째인지

, MONTHNAME(NOW()) // 이번 달 이름

, WEEKDAY(NOW()); // 몇 번째 요일(월요일:0)

SELECT CAST('1' AS UNSIGNED) // 문자를 숫자로 캐스팅(변경)

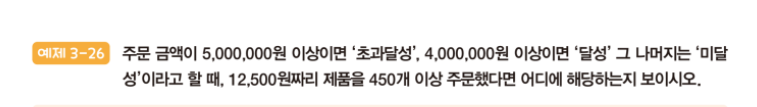
, CAST(2 AS CHAR(1)) // 숫자를 문자로

, CONVERT('1' , UNSIGNED) // 문자를 숫자로

, CONVERT(2 , CHAR(1)); // 숫자를 문자로

**IF(조건식, 참, 거짓)**

SELECT IF(1 \* 4 > 50, '참', '거짓');



SELECT CASE WHEN 12500\*450 > 5000000 THEN '초과달성'

WHEN 12500\*450 > 4000000 THEN '달성'

ELSE '미달성'

END;

**집계 함수**

Select 집계함수 from 테이블명 where 조건;

SELECT sum(마일리지) as 마일리지합 FROM 고객 where 도시='대전광역시';



Count(\*) : null 값 상관없이 모든 개수 셈

SELECT sum(마일리지), avg(마일리지), min(마일리지), max(마일리지) FROM 고객;

SELECT 도시, count(\*) as 고객수, avg(마일리지) as 평균마일리지 FROM 고객 **group by 도시**; -> **도시별로 도시 개수와 마일리지 평균 구하기**

SELECT 도시, count(\*) as 고객수, avg(마일리지) as 평균마일리지 FROM 고객 **group by 1;**

* Group by 1 써도 도시로 구분됨

SELECT 담당자직위, 도시, count(\*) as 고객수, avg(마일리지) as 평균마일리지 FROM 고객 group by 담당자직위, 도시 order by 1,2;

**예제 4-6 고객 테이블에서 도시별로 그룹으로 묶어서 고객 수와 평균마일리지 구하고, 고객수가 10명 이상인 레코드를 구하시오.**

SELECT 도시, count(\*) as 고객수, avg(마일리지) as 평균마일리지

FROM 고객 group by 도시 order by 1 **having count(\*) >=10;**

WITH ROLLUP : 그룹별 소계 구할 때 사용



SELECT IFNULL(도시, "총계") AS 도시, COUNT(\*) AS 고객수, AVG(마일리지) AS 평균마일리지

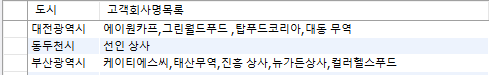
FROM 고객 WHERE 지역 IS NULL GROUP BY 도시 WITH ROLLUP;

SELECT 지역, count(\*) as 고객수, **grouping(지역) as 구분** from 고객 where 담당자직위='대표 이사' group by 지역 **with rollup**;

* Grouping(지역) as 구분: with rollup의 결과로 나온 null에 대해 1을 반환. 그렇지 않으면 0

SELECT GROUP\_CONCAT(이름) FROM 사원; -> 옆으로 나열

SELECT 도시, group\_concat(고객회사명) AS 고객회사명목록 FROM 고객 group by 1;



**Join**

조인(join) : 두 개 이상의 테이블을 연결하여 데이터를 검색하는 방법. 서로 다른 테이블에서 저장된 관련된 데이터를 가져와 하나의 결과로 표시한다. 검색하고 싶은 컬럼이 서로 다른 테이블에 있을 때 사용할 수 있다.

**FOREIGN KEY**

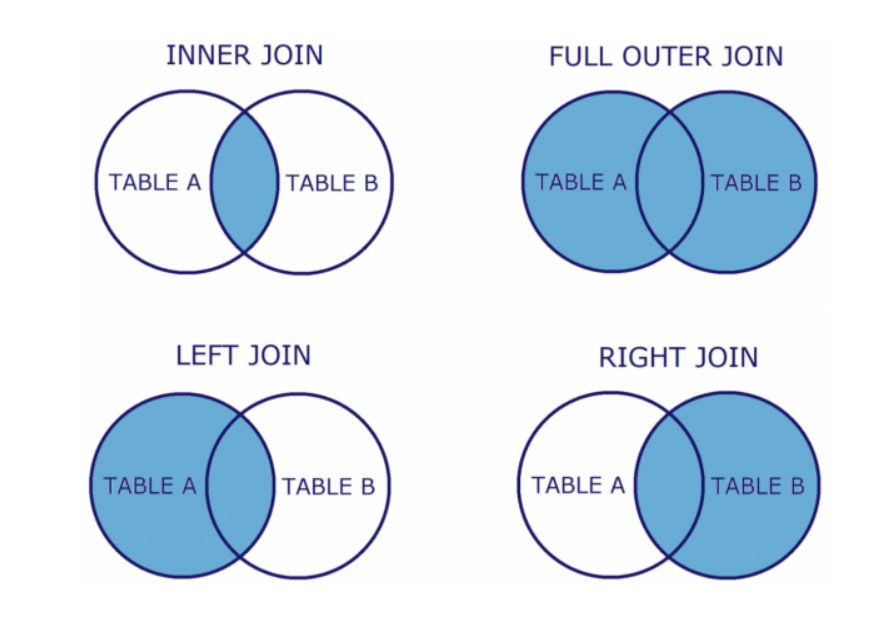
FOREIGN KEY(mem\_id) REFERENCES member(mem\_id)

* # 위의 member테이블에서 mem\_id와 엮어서 Foreign key 생성
* Member 테이블 : 부모, Key 테이블 : 자식
* 자식은 부모에게 없는 값을 가질 수 없다.

**INNER JOIN**

Ex) SNL\_SHOW 호스트로 출연한 celeb을 기준으로 celeb table과 snl\_show table을 inner join하여 각 table의 id와 이름 조회

SELECT celeb.id, celeb.name, snl\_show.id, snl\_show.host FROM celeb **INNER JOIN** snl\_show ON celeb.name = snl\_show.host;



**LEFT JOIN**

두 개의 테이블에서 공통 영역을 포함하여 왼쪽 테이블의 데이터를 조회하는 조인 방식

SELECT celeb.id, celeb.name, snl\_show.id, snl\_show.host FROM celeb **LEFT JOIN** snl\_show ON celeb.name = snl\_show.host;

**FROM : 테이블A , JOIN : 테이블B**

**FULL JOIN** : SQL에서 지원하지 않음

* LEFT JOIN과 RIGHT JOIN을 UNION하여 사용

SELECT celeb.id, celeb.name, snl\_show.id, snl\_show.host FROM celeb LEFT JOIN snl\_show ON seleb.name = snl\_show.host

**UNION**

SELECT celeb.id, celeb.name, snl\_show.id, snl\_show.host FROM celeb RIGHT JOIN snl\_show ON celeb.name = snl\_show.host;

**SELF JOIN**

INNER JOIN과 유사하고 TABLE 간의 공통된 데이터를 조회하며 가장 자주 쓰는 JOIN 방법

SELECT celeb.id, celeb.name, snl\_show.id, snl\_show.host FROM celeb, snl\_show WHERE celeb.name = snl\_show.host;

INNER JOIN -> , / ON -> WHERE

Ex) celeb table에서 snl\_show에 출연했고 소속사가 안테나인 셀럽의 이름과 직업 조회

1. LEFT JOIN(ANSI)

SELECT celeb.name, celeb.job\_title FROM celeb LEFT JOIN snl\_show ON celeb.name = snl\_show.host WHERE celeb.agency = ‘안테나’;

1. SELF JOIN(NON ANSI)

SELECT celeb.name, celeb.job\_title FROM celeb, snl\_show where celeb.name = snl\_show.host and celeb.agency = '안테나';

**UNION**

여러 개의 SQL문을 합쳐서 하나의 SQL문으로 만드는 방법(컬럼의 개수가 같아야 함)

중복값 제외됨

SELECT \* FROM test1 UNION SELECT \* FROM test2;

성별이 여자인 데이터를 검색하는 쿼리와 소속사가 YG엔터테인먼트인 데이터를 검색하는 쿼리를 union으로 실행 조회

SELECT \* FROM celeb WHERE sex='F' UNION SELECT \* FROM celeb WHERE agency LIKE 'YG%';

UNION ALL -> 중복 포함

SELECT c.이름, o.주문ID FROM 고객 c INNER JOIN 주문 o ON c.고객ID = o.고객ID;

* 테이블명을 임의로 지정할 수 있음. 고객: C, 주문: O

**서브쿼리**

메인쿼리 안에 또 다른 쿼리(서브쿼리)를 사용하여 데이터를 추출하거나 처리(select문 2개)

* 데이터 필터링/집계 및 계산/비교 및 검증/중첩된 데이터 추출/Exists절 : 특정조건을 만족하는 레코드 존재여부

**예제6-1. 최고 마일리지를 보유한 고객 정보를 보이시오.**

Sum()과 같은 집계함수는 where절에 직접 사용할 수 없고 반드시 서브쿼리를 통해 결과를 획득

SELECT 고객번호, 고객회사명, 담당자명, 마일리지 FROM 고객 **WHERE 마일리지 = (SELECT MAX(마일리지) FROM 고객);**

SELECT 고객번호, 고객회사명, 담당자명, 마일리지 FROM 고객 **ORDER BY 마일리지 DESC LIMIT 1;**

**예제6-2. 주문번호 ‘H0250’을 주문한 고객에 대해 고객회사명과 담당자명을 INNER JOIN과 서브쿼리로 구해보자.**

SELECT 주문번호, 주문.고객번호, 고객회사명, 담당자명 FROM 한빛무역.주문 **INNER JOIN** 고객 ON 주문.고객번호 = 고객.고객번호 WHERE 주문번호='H0250';

SELECT 주문번호, 주문.고객번호, 고객회사명, 담당자명 FROM 고객, 주문 WHERE 고객.고객번호 = 주문.고객번호 AND 주문.주문번호 **= (SELECT 주문.주문번호 FROM 주문 WHERE 주문번호='H0250');**

**예제6-3. ‘부산광역시’ 고객의 최소 마일리지보다 더 큰 마일리지를 가진 고객 정보를 보이시오.**

SELECT 담당자명, 고객회사명, 마일리지 FROM 고객 WHERE 마일리지 > (SELECT MIN(마일리지) FROM 고객 WHERE 도시 = '부산광역시');

**복수행 서브쿼리**

서브쿼리의 결과가 여러 행이 나오는 쿼리

(IN, ALL, ANY, SOME, EXISTS)와 같은 복수행 비교연산자를 사용하여 메인쿼리와 서브쿼리 연결

IN - 서브쿼리 결과에 해당 값이 포함되어 있는지

ALL - 서브쿼리의 결과 값과 비교해서 모든 조건을 만족해야 함

ANY/SOME – 서브쿼리 결과 어느 하나라도 조건을 만족하면 TRUE

EXISTS – 서브쿼리의 결과가 존재하기만 하면 TRUE. 보통 WHERE 절 내부에서 존재 여부 확인에 사용

**예제 1. ‘부산광역시’ 고객이 주문한 주문건수를 보이시오.**

SELECT COUNT(\*) AS 주문건수 FROM 주문 WHERE 고객번호 **IN** (SELECT 고객번호 FROM 고객 WHERE 도시='부산광역시');

**예제6-3. ‘부산광역시’ 전체 고객의 마일리지보다 마일리지가 큰 고객의 정보를 보이시오.**



SELECT 담당자명, 고객회사명, 마일리지 FROM 고객 WHERE 마일리지 > **ANY**(SELECT 마일리지 FROM 고객 WHERE 도시='부산광역시');

ANY와 ALL 차이점을 확인해보기

**예제6-6. 전체 지역 평균 마일리지보다 마일리지가 큰 고객의 정보를 보이시오.**

SELECT 담당자명, 고객회사명, 마일리지 FROM 고객 WHERE 마일리지 > **ALL**(SELECT AVG(마일리지) FROM 고객 GROUP BY 지역);

**예제6-7. 한 번이라도 주문한 고객의 정보를 보이시오.**

SELECT 고객번호, 고객회사명 FROM 고객 WHERE **EXISTS** (SELECT \* FROM 주문 WHERE 고객번호 = 고객.고객번호); -> EXISTS : 중복제거 됨

SELECT **DISTINCT** 고객.고객번호, 고객회사명 FROM 고객 **INNER JOIN** 주문 ON 고객.고객번호=주문.고객번호;

**조건절에서 사용하는 서브쿼리**

HAVING : GROUPING 후 조건을 거는 것. 집계값 필터링

WHERE : GROUPING 전에 조건을 거는 것. 일반행 필터링

**예제6-8. 평균 마일리지가 고객 전체의 평균 마일리지보다 큰 도시에 대해 도시명과 도시의 평균 마일리지를 보이시오.**

SELECT 도시, AVG(마일리지) AS 평균마일리지 FROM 고객 GROUP BY 도시 HAVING AVG(마일리지) > (SELECT AVG(마일리지) FROM 고객);

* HAVING AVG(마일리지) -> 도시 평균 마일리지
* SELECT ~ -> 전체 평균 마일리지

**FROM절에서 사용하는 서브쿼리**

FROM절에서도 서브쿼리를 사용할 수 있다. 이를 인라인 뷰라고 한다. 인라인 뷰에는 반드시 별명을 지정해주어야 하며, 인라인 뷰의 별명은 테이블명처럼 사용할 수 있다.

**예제6-9. 담당자명, 고객회사명, 마일리지, 도시, 해당 도시의 평균 마일리지를 보이시오. 그리고 고객이 위치하는 도시의 평균 마일리지와 각 고객의 마일리지 간의 차이도 함께 보이시오.**



**SELECT문**

SELECT 담당자명, 고객회사명, 마일리지, 고객.도시, 도시\_평균마일리지, (도시\_평균마일리지-마일리지) AS 차이 FROM 고객, (SELECT 도시, AVG(마일리지) AS 도시\_평균마일리지 FROM 고객 GROUP BY 도시) AS 도시별요약 WHERE 고객.도시 = 도시별요약.도시;

**WITH문**

WITH 도시별요약 AS (SELECT 도시, AVG(마일리지) AS 도시\_평균마일리지 FROM 고객 GROUP BY 도시) SELECT 담당자명, 고객회사명, 마일리지, 고객.도시, 도시\_평균마일리지, 도시\_평균마일리지-마일리지 AS 차이 FROM 고객, 도시별요약 WHERE 고객.도시 = 도시별요약.도시;

**SELECT절에서 사용하는 서브쿼리** : 스칼라 서브쿼리. 정확히 한 개의 행에 대해 한 개의 값을 반환

**예제6-10. 고객번호, 담당자명, 고객의 최종 주문일을 보이시오.**

SELECT 고객번호, 담당자명, (SELECT MAX(주문일) FROM 주문 WHERE 주문.고객번호 = 고객.고객번호) AS 최종주문일FROM 고객;

SELECT 고객.고객번호, 담당자명, MAX(주문.주문일) AS 최종주문일 FROM 고객 **LEFT JOIN** 주문 ON 고객.고객번호 = 주문.고객번호 GROUP BY 고객.고객번호, 고객.담당자명;

CTE는 뷰와 마찬가지로 파생 테이블처럼 사용 가능

WITH CTE명 AS (SELECT ~ )

**상관 서브쿼리**

메인쿼리와 서브쿼리 간의 상관관계를 포함하는 형태의 쿼리

**사원테이블에서 사원번호, 사원의 이름, 상사의 사원번호, 상사의 이름을 보이시오.**

SELECT 사원번호, 이름, 상사번호, (SELECT 이름 FROM 사원 AS 상사 WHERE 상사.사원번호 = 사원.상사번호) AS 상사이름 FROM 사원;

**SELF JOIN**

같은 테이블을 JOIN 함. JOIN한 테이블에 별칭을 줌으로 구분함.

LEFT JOIN으로 상사번호가 없는 사원도 함께 표시

SELECT 사원.사원번호, 사원.이름, 사원.상사번호, 상사.이름 AS 상사이름 FROM 사원 LEFT JOIN 사원 AS 상사 ON 사원.상사번호 = 상사.사원번호;

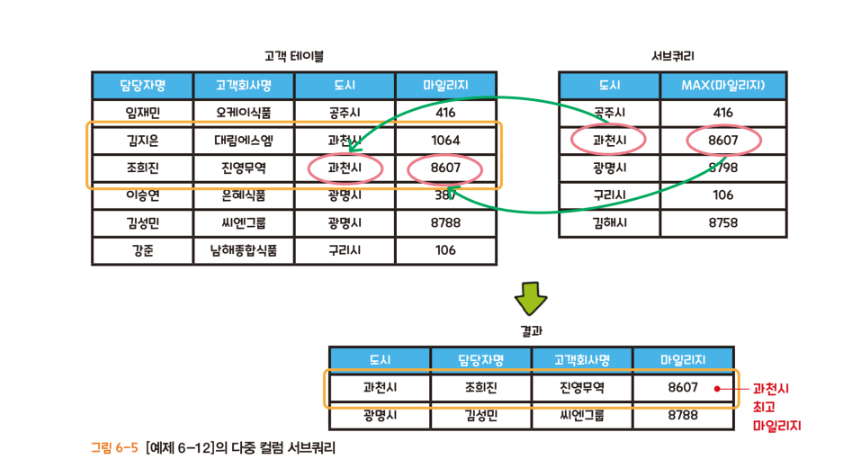
**다중 컬럼 서브 쿼리**

여러 개의 컬럼을 사용하여 다중 비교하는 쿼리

문제. 각 도시마다 최고 마일리지를 보유한 고객의 정보를 보이시오.

SELECT 도시, 담당자명, 고객회사명, 마일리지 FROM 고객 WHERE (도시, 마일리지) IN (SELECT 도시, MAX(마일리지) FROM 고객 GROUP BY 도시);





SQL문의 종류

1. 데이터 정의어(DDL, Data Definition Language)

데이터 베이스나 테이블, 인덱스와 같은 데이터베이스 객체를 생성, 변경, 삭제할 때 쓰임

* Create : 데이터베이스 객체 생성
* Alter : 생성된 객체의 구조 변경
* Drop : 생성된 객체를 삭제
* Truncate : 테이블에 있는 데이터 모두 삭제

1. 데이터 조작어(DML, Data Management Language)

* 데이터 관리. 레코드 추가, 삭제, 변경할 때 쓰임
* Select : 테이블에서 조건에 맞는 데이터 검색
* Insert : 테이블에 새로운 레코드 삽입
* Update : 테이블에 있는 데이터 값을 수정
* Delete : 테이블에 있는 데이터 삭제

1. 데이터 제어어(DCL)

* GRANT : 특정 사용자 또는 특정 객체에 대해 생성, 수정 등 특정 작업에 권한 부여
* Revoke : grant문으로 부여한 권한을 철회할 때 사용

1. TCL

* Commit : 트랜잭션 작업 내용을 실제 데이터베이스에서 영구 저장
* Rollback : 트랜잭션 처리과정에서 발생한 변경 사항을 취소