서브쿼리 응용 – 테이블 생성

테이블 구조와 데이터 복사

```
create table emp_copy
select * from employees;
```

테이블 구조만 복사

```
create table emp_blank
select * from employees
where 1 = 0;
```

서브쿼리 응용 – 추가 수정 삭제

```
insert into emp_blank
select * from employees
where department_id = 80;
```

```
update emp50
set sal = sal + 500
where sal < (select avg(salary) from employees);
```

```
delete from emp50
where sal < (select avg(salary) from employees);
```

MySQL자료형

숫자

데이터 형식	바이트 수	타입	설명
TINYINT	1	정수	-128 ~ 127
SMALLINT	2	정수	-32,768 ~ 32,767
INT(INTEGER)	4	정수	약 -21억 ~ 21억
BIGINT	8	정수	약 -900경 ~ 900경
FLOAT	4	실수	-3.40E+38 ~ 1.17E-38
DECIMAL(m.[d])	5~17	실수	-10E38+1 ~ 10E38-1
NUMERIC(m.[d])	-	-	전체자릿수(m), 소숫점자릿수(d)

MySQL자료형

문자

데이터 형식	바이트 수	설명
CHAR(n)	1~255	고정길이 문자형, CHAR만 쓰면 CHAR(1)과 동일
VARCHAR(n)	1~65535	가변길이 문자형, 메모리 절약에 좋음
LONGTEXT	1~4294967295	최대 4GB 크기의 TEXT 데이터 값
LONGBLOB	1~4294967295	최대 4GB 크기의 BLOB 데이터 값

MySQL자료형

날짜

데이터 형식	바이트 수	형식	범위
DATE	3	'YYYY-MM-DD'	1001-01-01 ~ 9999-12-31
TIME	3	'HH:MM:SS'	-838:59:59.000000 ~ 838:59:59.000000
DATETIME	8	'YYYY-MM-DD HH:MM:SS'	1001-01-01 00:00:00 ~ 9999-12-31 23:59:59

테이블 생성

- create table

```
> create table table_name (
        column_name1 Type [optional attributes],
        column_name2 Type,
        .
        column_nameN Type,
    );
```

- optional attributes
  - NOT NULL : 각 행은 해당 열의 값을 포함해야 하며 null값은 허용되지 않음.
  - DEFAULT value : 값이 전달되지 않을 때 추가되는 기본값 설정.
  - UNSIGNED : Type이 숫자인 경우만 해당되며 숫자가 0 또는 양수로 제한됨.
  - AUTO INCREMENT : 새 레코드가 추가 될 때마다 필드 값을 자동으로 1증가시킴.
  - PRIMARY KEY : 테이블에서 행을 고유하게 식별하기 위해 사용. PRIMARY KEY 설정이 있는 열은 일반적으로 ID번호이며 AUTO INCREMENT와 같이 사용되는 경우가 많음.

테이블 생성 - 제약조건

- 제약 조건
  - 컬럼에 저장될 데이터의 조건을 설정하는 것.
  - 제약조건을 설정하면 조건에 위배되는 데이터는 저장 불가.
  - 테이블 생성시 컬럼에 직접 지정하거나 constraint로 지정, 또는 ALTER를 이용하여 설정가능.

제약 조건	description
NOT NULL	컬럼에 NULL값을 저장할 수 없고, 반드시 쿼리문을 이용하여 값을 지정.
UNIQUE	컬럼에 중복된 값을 저장 할 수 없음, NULL값은 허용.
PRIMARY KEY	컬럼에 중복된 값을 저장 할 수 없고, NULL값도 허용하지 않음. 주로 ROW를 구분하기 위한 유일한 값을 지정할 때 사용. '기본키'라고도 부름.
FOREIGN KEY	특정 테이블의 PK 컬럼에 저장되어 있는 값만 저장. '참조키', '외래키'라고도 부름. NULL값은 허용. references를 이용하여 어떤 컬럼에 어떤 데이터를 참조하는지 반드시 지정.
DEFAULT	NULL값이 들어올 경우 기본 설정되는 값을 지정.
CHECK	값의 범위나 종류를 지정. MYSQL 8.0.16부터 사용가능. (이전 버전의 경우 설정은 되나 적용이 안됨)

테이블 변경

- 컬럼 추가

```
ALTER TABLE 테이블명 ADD COLUMN 컬럼명 data-type 제약조건;
```

- 컬럼 변경

```
ALTER TABLE 테이블명 MODIFY COLUMN 컬럼명 data-type 제약조건;
```

- 컬럼 이름 변경

```
ALTER TABLE 테이블명 CHANGE COLUMN 이전컬럼명 변경컬럼명 data-type 제약조건;
```

- 컬럼 삭제

```
ALTER TABLE 테이블명 DROP COLUMN 컬럼명;
```

- 테이블 이름 변경

```
ALTER TABLE 이전테이블명 RENAME 변경테이블명 ;
```

View

- 데이터베이스에 존재하는 '가상의 테이블'.
- 실제 행과 열을 가지고 있지만 데이터를 저장하고 있지는 않음.
- 테이블처럼 물리적으로 저장되는 것은 아님.
- join이나 subquery와 같이 여러 개의 테이블을 참조하여 데이터를 조회 할 때 번거로움을 줄일 수 있음.
- 단, MySQL에서 View는 단지 다른 테이블이나 View에 있는 데이터를 보여주는 역할만 수행.
- View와 Table의 차이점은, Table은 실질적인 데이터가 있지만 View는 데이터가 없고 SQL만 저장한다.

View

✓ View 장점

- 특정 사용자에게 테이블 전체가 아닌 필요한 필드만(보안성) 보여줄 수 있음.

  >> DBMS의 사용자별 권한 관리 기능을 통해 사용자가 테이블에 직접적인 접근을 하지 못하도록 막을 수 있음.

- 복잡한 쿼리를 단순화해서 사용할 수 있음.
- 쿼리를 재사용할 수 있음.
- 여러 방법의 데이터 조회에 알맞은 다양한 구조의 데이터 분석 기반을 구축할 수 있음.

  >> 기존 테이블 구조를 변경하지 않음.

✓ View 단점

- 삽입, 삭제, 갱신 작업에 많은 제한 사항을 가짐.
- View는 자신만의 인덱스를 가질 수 없음.

View 생성

- create 문 사용.

```
create view 뷰이름
as
select 필드이름1, 필드이름2, ...
from 테이블이름
where 조건;
```

- select 문에서 선택된 필드를 이용하여 새로운 View를 생성.
- View는 원본 테이블의 이름과 같은 이름을 사용 할 수 없음.

- 설정한 필드를 대체하기 위해서는 새로운 View로 대체해야 함.

```
create or replace view 뷰이름
as
select 필드이름1, 필드이름2, ...
from 테이블이름
where 조건;
```

- 뷰이름에 해당하는 View가 존재하면 replace, 존재하지 않으면 create가 일어 남.

View 수정과 삭제

✓ View 수정

- alter 사용.

```
alter view 뷰이름
as
select 필드이름1, 필드이름2, ...
from 테이블이름;
```

✓ View 삭제

- drop 사용.

```
drop view 뷰이름;
```