REM Author : Henry
REM Date : 2024.06.12
REM Objective :
REM Environment : Ubuntu Server 22.04 LTS, MySQL Workbench 8.0 CE, MySQL Community Server 8.0.37-0ubuntu0.22.04.3 (ubuntu)

REM DDL(Data Definition Language)
-데이타베이스의 Object 구조를 생성, 변경, 삭제하는 명령어
-CREATE, ALTER, DROP, RENAME, TRUNCATE

REM Naming Convension
출처: https://taptorestart.tistory.com/entry/MySQL-데이터베이스명-테이블명-컬럼명은-어떻게-지어야-할까

1. General
1)Ensure the name is unique and does not exist as a reserved keyword.
2)Keep the length to a maximum of 30 bytes—in practice this is 30 characters unless you are using a multi-byte character set.
3)Names must begin with a letter and may not end with an underscore.
4)Only use letters, numbers and underscores in names.
5)Avoid the use of multiple consecutive underscores—these can be hard to read.
6)Use underscores where you would naturally include a space in the name (first name becomes first_name).
7)Avoid abbreviations and if you have to use them make sure they are commonly understood.

2. Tables
1)Use a collective name or, less ideally, a plural form. For example (in order of preference) staff and employees.
2)Do not prefix with tbl or any other such descriptive prefix or Hungarian notation.
3)Never give a table the same name as one of its columns and vice versa.
4)Avoid, where possible, concatenating two table names together to create the name of a relationship table. Rather than cars_mechanics prefer services.

3. Columns
1)Always use the singular name.
2)Where possible avoid simply using id as the primary identifier for the table.
3)Do not add a column with the same name as its table and vice versa.
4)Always use lowercase except where it may make sense not to such as proper nouns.

REM Database 생성
1. Database
-Table이나 View, Index 등 데이터베이스 내에 정의하는 모든 것
2. The CREATE DATABASE statement is used to create a new SQL database.
3. Syntax
```
CREATE DATABASE [IF NOT EXISTS] db_name
[create_option] ...

create_option: [DEFAULT] {
```

```
    CHARACTER SET [=] charset_name | COLLATE [=] collation_name
  }

4. Example
  CREATE DATABASE testDB;
  CREATE DATABASE study_db default CHARACTER SET UTF8;


REM Database 삭제
1. The DROP DATABASE statement is used to drop an existing SQL database.
2. Syntax
  DROP DATABASE [IF EXISTS] db_name
3. Example
  DROP DATABASE testDB;



REM 테이블의 생성
1. CREATE TABLE
  -The CREATE TABLE statement is used to create a new table in a database.

2. Syntax
    CREATE TABLE [IF NOT EXISTS] table_name
    (
      column_name     column_definition
      [CONSTRAINT] PRIMARY KEY |
      [CONSTRAINT] UNIQUE |
      [CONSTRAINT] FOREIGN KEY |
      CHECK (expression)
    )
    column_definiction : {
      data_type [NOT NULL | NULL] [DEFAULT default_value]
        [AUTO_INCREMENT] [UNIQUE][PRIMARY KEY]
        [COMMENT 'string']
    }
    table_option : {
      AUTO_INCREMENT [=] value |
      [DEFAULT] CHARACTER SET [=] charset_name |
      COMMENT [=] 'string'
    }

  1)일반적 생성 방법
    CREATE TABLE table_name ( column_name     datatype, ...);

    CREATE TABLE IF NOT EXISTS dept1
    (
      deptno    TINYINT,
      dname     VARCHAR(13),
      loc       VARCHAR(14) DEFAULT 'Seoul'
    )DEFAULT CHARACTER SET UTF8
```

2)SUB_QUERY 에 의해 검색된 테이블과 동일한 구조로 생성
-테이블 복사, CTAS(Create Table ~ As Select ~)방법
-주의할 점은 기존 테이블의 제약조건이 모두 없어진다는 점

```
CREATE TABLE dept2
AS
SELECT * FROM dept;

CREATE TABLE emp2
AS
SELECT EMPNO, ENAME FROM EMP;

CREATE TABLE emp3
AS
SELECT * FROM emp
WHERE DEPTNO = 10;

CREATE TABLE emp4
AS
SELECT * FROM emp where 1=0;  <-- 구조만 복사
```

3)Guide Lines
-지정한 열 이름을 사용하여 테이블을 생성하며 SELECT 문에 의해 검색된 행을 테이블에 삽입.
-열 사양이 제공되는 경우 열 수는 하위 질의 SELECT 목록의 열 수와 동일해야 한다.
-열 사양이 제공되지 않는 경우 해당 테이블의 열 이름은 하위 질의의 열 이름과 동일.

4)조회

```
SELECT table_name
FROM user_tables;

SELECT object_name, object_type
FROM user_objects;

CREATE TABLE dept30
AS
SELECT empno, ename, sal * 12 "Annual Salary", hiredate
FROM emp
WHERE deptno = 30;
```

5)사용예

```
--부서별로 인원수, 평균급여, 급여의 합, 최소급여, 최대급여를 포함하는 emp_calcu table을 생성하라.
CREATE TABLE emp_calcu
AS
SELECT deptno, COUNT(*) AS cnt, AVG(sal) AS salavg, SUM(sal) AS salsum,
            MIN(sal) AS minsal, MAX(sal) AS maxsal
FROM emp
GROUP BY deptno;
```

```
--사원번호, 이름, 업무, 입사일자, 부서번호만 포함하는 emp_temp table을 생성하는데, 자료는 포함하지 않고 스키마만 생성하시오.
CREATE TABLE emp_temp
AS
SELECT empno, ename, job, hiredate, deptno
FROM emp
WHERE 1 < 0;


--테이블을 다음챠트를 기반으로 DEPARTMENT 테이블을 생성하고, 생성한 후 테이블 구조를 확인하시오.
--열이름 :              id              name
--데이타유형 :          INT          VARCHAR
--길이 :                  7                25
CREATE TABLE department
(
    id        INT(7),
    name      VARCHAR(25)
);

DESC department


--테이블을 다음챠트를 기반으로 EMPLOY 테이블을 생성하고, 생성한 후 테이블 구조를 확인하시오.
--열이름 :       id            last_name      first_name      dept_id
--데이타유형 :   INT           VARCHAR        VARCHAR         INT
--길이 :          7              25               25              7
CREATE TABLE employ
(id       INT(7),
    last_name      VARCHAR(25),
    first_name   VARCHAR(25),
    dept_id       INT(7)
);

DESC employ



REM TABLE 삭제
1. The DROP TABLE statement is used to drop an existing table in a database.
   -table의 모든 구조와 데이터가 삭제
   -DDL이기 때문에 TRANSACTION 이 COMMIT된다.
   -영구히 삭제, 되돌릴 수 없다.
2. Syntax
    DROP TABLE table_name;

    DROP TABLE emp30;
    DROP TABLE dept;

3. Guideline
```

1)삭제하려는 테이블의 기본 키나 고유 키를 다른 테이블에서 참조해서 사용하는 경우에는 해당 테이블을 제거할 수 없다.
2)이런 경우에는 참조하는 테이블을 먼저 제거한 후 해당 테이블을 삭제해야 한다.

REM **ALTER TABLE**
1. The **ALTER TABLE statement is** used **to add, delete, or modify** columns **in** an existing **table.**
2. The **ALTER TABLE statement is** also used **to add and drop** various **constraints on** an existing **table.**
-테이블의 스키마 변경
-**Column**을 추가/삭제하거나 제**약**조건을 추가/삭제하는 작업

3. **Column Add**
-열 추가
-새 열은 마지막 열, 위치 지정 불가능
-이미 행을 포함하고 있는 테이블이**라**면 새로 들어갈 열의 모든 행은 초기에 널 값을 갖는다.
-Syntax

```
ALTER TABLE table_name
ADD  [COLUMN] col_name column_definition
    [FIRST | AFTER col_name]
        [, column datatype]...);

ALTER TABLE dept30
ADD job VARCHAR(9);

ALTER TABLE dept30
ADD sal INT AFTER ename;

ALTER TABLE dept1
ADD bigo   VARCHAR(15) FIRST;
```

2. **Column Modify**
-**Column**의 데이터 유형, **Default**값, **NOT NULL** 제**약**조건에 대한 변경
-Syntax

```
ALTER TABLE table_name
MODIFY  [COLUMN] column_name column_definition
    [FIRST | AFTER col_name]
```

-**MODIFY** Guide Lines

```
--숫자 열의 너비 또는 전체 자릿수를 증가가능.
--열이 널 값만 포함하고 테이블에 행이 없는 경우 열의 너비를 줄일 수 있다.
--열이 널 값을 포함하면 데이터 유형을 변경할 수 있다.
--열의 기본값을 변경하면 변경 이후에 테이블에 삽입되는 항목에만 영향을 준다.
--테이블의 구조 변경(데이터 타입, 길이), 만일 기존 데이터가 있을 경우에는 CHAR와 VARCHAR 사이의 타입 변경만 가능
--컬럼의 크기 변경 역시 기존에 저장된 데이터의 길이와 같거나 클 경우에만 변경이 가능
--즉 해당 컬럼에 자료가 없을 경우에는 데이터타입과 컬럼의 크기를 변경가능
--해당 컬럼에 자료가 있을 경우에는 데이터타입 변경 불가능하고 크기는 늘릴 수만 있음.
```

```
        ALTER TABLE dept30
        MODIFY ename VARCHAR(15);

        ALTER TABLE dept1
        MODIFY bigo   VARCHAR(30);


    -사용예
        --위 예제에서 생성한 EMPLOY테이블의 last_name열에 긴 성을 가진 사원의 성을 저장할 수 있도록 길이를 50으로 수정한 후, 내용을 확인하시오


3. Column Remove
    -열 삭제
    -Guide Lines
        --열은 데이터를 포함하거나 포함하지 않거나 삭제 할 수 있다.
        --한 번에 하나의 열만 삭제할 수 있다.
        --테이블에는 최소 하나의 열이 있어야 하므로 열을 삭제하려면 테이블에 열이 둘 이상 있어야 한다.
        --삭제된 열은 복구할 수 없다.
    -Syntax
        ALTER TABLE table_name
        DROP [COLUMN] col_name

        ALTER TABLE emp1
        DROP COLUMN JOB;


4. CHANGE COLUMN
    -Column의 이름을 변경
    -Syntax
        ALTER TABLE table_name
        CHANGE [COLUMN] old_col_name new_col_name column_dinifition
            [FIRST | AFTER col_name]

        ALTER TABLE dept1
        CHANGE COLUMN bigo bigo1 VARCHAR(15);


REM TABLE TRUNCATE
1. Empties a table completely.
2. ROLLBACK 불가능
3. DELETE는 모든 행을 제거할 수 있지만, 저장공간을 해제할 수 없다.
4. 모든 행을 제거할 때는 DELETE보다 TRUNCATE TABLE문을 사용하자.
5. Syntax
        TRUNCATE [TABLE] table_name

        TRUNCATE TABLE emp30;
```

6. **DROP TABLE**은 테이블 자체**를** 제거하지만**, TRUNCATE**는 테이블은 존재하면서 데이터만 제거하기에 구조는 남아있다**.**

REM **Rename TABLE**
1. **table**의 이름 변경
2. Syntax
   RENAME TABLE old_table_name **TO** new_table_name

   RENAME TABLE dept2 **TO** dept3**;**

REM 테이블에 주석 문 추가
1. **Table Comment**
   -CREATE TABLE 또는 **ALTER TABLE**에 table_option**으**로 추가
   COMMENT [=] 'string'

   ALTER TABLE dept1
   COMMENT = '부서 정보 테이블입니다'**;**

2. **Column Comment**
   -CREAET **TABLE** 또는 **ALTER TABLE**에 column_definition**으**로 추가
   COMMENT 'string'

   ALTER TABLE dept1
   MODIFY COLUMN deptno TINYINT **COMMENT** '부서코드'

REM **DATA TYPE**
1. CHAR
   1**)** 고정길이의 문자 데이타
   2**)** 기본값 및 최소크기 **:** 1Byte
   3**)** 최대크기 255Bytes
   4**)** 나머지 공간을 여백**으**로 채워서 처음 정의된 공간을 모두 사용하는 타입

2. VARCHAR
   1**) Variable Character**의 **약**자
   2**)** 가변길이의 문자 데이타
   3**)** 기본값 및 최소 크기 **:** 1Byte
   4**)** 최대크기 **:** 65535Bytes
   5**)** 여백**으**로 채우지 않고 필요한 공간만 사용

   ABCDE --> char(8) --> ABCDE_ _ _
   ABCDE --> varchar(8) --> ABCDE

3. Text Data
   1) VARCHAR 열에 64KB 제한을 초과하는 데이터를 저장하려면 텍스트 자료형 중 하나를 사용해야 한다.
   2) MySQL Text Type
      -TINYTEXT : 255
      -TEXT : 65,535
      -MEDIUMTEXT : 16,777,215
      -LONGTEXT : 4,294,967,295
   3) 고려사항
      -텍스트 열에 로드되는 데이터가 해당 유형의 최대 크기를 초과하면 데이터가 잘린다.
      -데이터를 열에 로드되면 후행 공백이 제거되지 않는다.
      -정렬 또는 그룹화에 TEXT 열을 사용할 경우, 필요하다면 한도를 늘릴 수 있지만, 처음에는 1,024 Bytes만 사용된다.
      -TEXT를 제외한 텍스트 자료형은 MySQL의 고유한 자료형이다. cf)Oracle CLUB
      -MySQL은 이제 VARCHAR 열에 최대 65,535 Bytes를 허용하므로(Version4에서는 255 Bytes로 제한됨) TINYTEXT나 TEXT 자료형을 사용할 필요가 없다.

4. 정수
   1) TINYINT(1), SMALLINT(2), INT(4), BIGINT(8)
   2) 부호없는 정수를 저장할 때는 UNSIGNED 예약어를 뒤에 붙여준다.
   3) MySQL 정수 자료형
      -TINYINT : -128 ~ 127 | 0 ~ 255
      -SMALLINT : -32,768 ~ 32,767 | 0 ~ 65,535
      -MEDIUMINT : -8,388,608 ~ 8,388,607 | 0 ~ 16,777,215
      -INT : -2,147,483,648 ~ 2,147,483,647 | 0 ~ 4,294,967,295
      -BIGINT : -2^63 ~ 2^63 - 1 | 0 ~ 2^64 - 1

5. 실수
   1) 숫자값을 -38자리 ~ 38자리(-308자리 ~ 308자리) 저장
   2) MySQL 부동소수점 자료형
      -FLOAT(p,s) : -3.402823466E+38 ~ -1.175494351E-38
                    1.175494351E-38 ~ 3.402823466E+38
      -DOUBLE(p,s) : -1.7976931348623157E+308 ~ -2.2250738585072014E-308
                     2.2250738585072014E-308 ~ 1.7976931348623157E+308
      -p(recision, 소수점 왼쪽과 오른쪽 모두에 허용되는 자릿수) : 정밀도
      -s(cale, 소수점 오른쪽의 허용 자릿수) : 척도
   3) 부동소수점 열에 정밀도와 척도를 지정할 때, 자릿수가 열의 척도 및(또는) 정밀도를 초과하면 열에 저장된 데이터는 반올림된다.
   4) 예를 들어, FLOAT(4,2) -> 27.44, 8.19의 경우 허용, 하지만 17.8675는 17.87로 반올림되며, 178.375를 저장하면 오류가 발생한다.

6. DATE
   1) MySQL 시간 자료형
      -DATE : YYYY-MM-DD | 1000-01-01 ~ 9999-12-31
      -DATETIME : YYYY-MM-DD HH:MI:SS | 1000-01-01 00:00:00.000000 ~ 9999-12-31 23:59:59.999999
      -TIMESTAMP : YYYY-MM-DD HH:MI:SS | 1970-01-01 00:00:00.000000 ~ 2038-01-18 22:14:07.999999
      -YEAR : YYYY | 1901 ~ 2155

-TIME : HHH:MI:SS | -838:59:59.000000 ~ 838:59:59.000000
2) 날짜 형식의 구성요소
-YYYY : 연도, 세기 포함 | 1000 ~ 9999
-MM : 월 | 01 ~ 12
-DD : 일 | 01 ~ 31
-HH : 시간 | 00 ~ 23
-HHH : 시간(경과) | -838 ~ 838
-MI : 분 | 00 ~ 59
-SS : 초 | 00 ~ 59
3) 사용자가 테이블의 특정 행을 마지막으로 수정한 시기를 추적하는 열에는 TIMESTAMP 자료형을 사용한다.
4) TIMESTAMP 자료형에는 DATETIME 자료형과 동일한 정보(년,월,일 시,분,초)가 저장되지만, 테이블에 행이 추가되거나 수정될 때 MySQL 서버에 의해 현재 날짜/시간으로 TIMESTAMP 열이 자동으로 채워진다.
5) 작업 완료까지 필요한 시간 데이터를 포함하는 열에는 TIME 자료형이 사용된다.
6) 이러한 유형의 데이터는 작업을 완료하는 데 필요한 시간/분/초에만 관심이 있으므로 날짜 구성요소를 저장할 필요는 없다.
7) 이 정보는 두 개의 DATETIME 열(작업 시작 날짜/시간 열, 작업 완료 날짜/시간 열)을 사용하여 하나를 다른 열에서 빼서 알아낼 수도 있지만, 한 개의 TIME 열을 사용하는 편이 더 간단하다.

REM **Character set**
1. 영어와 같이 Latin 알파벳을 사용하는 언어는 각 문자를 저장하기 위해 단 1Byte만 필요
2. 하지만, 한국어나 일본어 및 다른 언어들은 많은 수의 문자를 포함하기 때문에 각 문자마다 여러 Byte의 저장 공간이 필요하다.
3. 이러한 문자 집합, 즉 **Character set**을 Multibyte **Character Sets** 라고 한다.
4. MySQL은 Single Byte 및 Multi Bytes의 다양한 **Character set**을 모두 사용하여 데이터를 저장할 수 있다.
5. 확인하는 방법은 다음과 같다.
```
mysql> SHOW CHARACTER SET;
```
6. 위의 결과에서 Maxlen의 값이 1보다 크면 **Character set**이 Multibyte **Sets**라는 의미이다.
7. 이전 버전의 MySQL 서버에서는 latin1 **Character set**이 기본 캐릭터셋으로 자동선택되었지만, Version 8에서는 기본적으로 utf8mb4로 설정된다.
8. 그러나, 데이터베이스의 각 문자열에 대해 다른 **Character set**을 사용하도록 선택할 수 있다.
9. 열을 정의할 때 기본값이 아닌 **Character set**을 선택하려면 다음과 같이 자료형 정의 뒤에 지원되는 **Character set** 중 하나를 지정한다.
```
mysql> VARCHAR(20) CHARACTER SET latin1;
```
10. MySQL을 사용하면 전체 데이터베이스에 대한 기본 **Character set**을 설정할 수도 있다.
```
mysql> CREATE DATABASE european_sales CHARACTER SET latin1;
```