SQL in MySQL

```
#database 생성
CREATE DATABASE db_test;

#database 확인
SHOW DATABASES;

#database 삭제
DROP DATABASE db_test;

#사용자 생성(id : db_min / pw : 1234qwer)
CREATE USER 'db_min'@'localhost' IDENTIFIED BY '1234qwer';

#사용자 삭제
DROP USER 'db_min'@'localhost';

#사용자 권한 주기(all)
GRANT ALL ON db_test.* to 'db_min'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;

#사용자 권한 없애기(all)
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'db_min'@'localhost';

#database 사용하기
USE db_test;

# table 만들기
CREATE TABLE user (
        seq INT,
        id VARCHAR(12),
        password VARCHAR(12)
);
# PRIMARY KEY 만들기
CREATE TABLE user (
        seq INT PRIMARY KEY,
        id VARCHAR(12),
        password VARCHAR(12)
);
# NOT NULL 설정
CREATE TABLE user (
        seq INT PRIMARY KEY,
        id VARCHAR(12) NOT NULL,
        password VARCHAR(12) NOT NULL
);
# UNIQUE 설정
CREATE TABLE user (
        seq INT PRIMARY KEY,
        id VARCHAR(12) NOT NULL UNIQUE
```

```
);

# DEFAULT 설정
CREATE TABLE user (
        seq INT PRIMARY KEY,
        id VARCHAR(12) NOT NULL,
        password VARCHAR(12) NOT NULL,
        level TINYINT NOT NULL DEFAULT 10
);
# 다중 UNIQUE or PRIMARY KEY 설정
CREATE TABLE user (
        seq INT,
        id VARCHAR(12) NOT NULL,
        password VARCHAR(12) NOT NULL
        PRIMARY KEY (seq, id),
        UNIQUE (id, password)
);
# AUTO_INCREMENT 설정
CREATE TABLE user (
        num INT AUTO_INCREMENT PRIMARY KEY,
        str1 VARCHAR(6)
);

# Table 생성 삭제 팁
DROP TABLE IF EXISTS user;
CREATE TABLE user (
        num INT AUTO_INCREMENT PRIMARY KEY,
        str1 VARCHAR(6)
);

#table 확인 1
DESC user;

#table 확인 2
SHOW CREATE TABLE user;

#table 삭제
DROP TABLE db_test;

#table에 데이터 삽입
INSERT INTO user (seq, id, password) VALUES (1, 'nobody', '1234qwer');
INSERT INTO user VALUES (1, 'nobody', '1234qwer');
INSERT INTO user (id, seq, password) VALUES ('nobody', 1, '1234qwer');

#table에 데이터 확인
SELECT * FROM user;
SELECT id FROM user;
SELECT id,password FROM user WHERE id = 'nobody';

#스키마 복사
CREATE TABLE user LIKE old_user;
```

#데이터 복사
```
INSERT INTO user (SELECT * FROM old_user) ;
```

#현재 시간 입력
```
INSERT INTO time_test(time1) VALUES ( now() ) ;
```

#이미지 파일 저장
```
INSERT INTO test_table (image) VALUES (load_file('c:\\test\\slime.png'));
```

#저장된 이미지 파일 파일로 가져오기
```
SELECT image INTO DUMPFILE 'c:\\test\\dump.png' FROM test_table
WHERE filename = 'slime.png';
```

#Data Import
1. CSV파일로 exprot
2. charset 확인
```
LOAD DATA LOCAL INFILE 'C:\\test\\db_ex2.csv'
INTO TABLE test FIELDS TERMINATED BY ',';
LOAD DATA LOCAL INFILE 'C:\\test\\db_ex2.csv'
```
3. 1줄 무시
```
INTO TABLE test FIELDS TERMINATED BY ',' IGNORE 1 LINES;
```

#Data Export
1. mysql server의 bin 디렉토리로 이동(터미널 이용)
2. $ `mysqldump –u id –p database_name table_name > file_name.sql`

tip. PATH 환경변수 설정

#SQL file Execute
1. 터미널 이용
2. $ `mysql –u id –p database_name < file_name.sql`

#SELECT Projection
```
SELECT id, name FROM user;
SELECT id, league FROM user;
```

#SELECT Selection
```
SELECT * FROM user WHERE lv=25;
SELECT * FROM user WHERE score > 2000;
```

#SELECT Selection & Projection
```
SELECT id, league FROM user WHERE score >= 2000 AND score <= 3000;
SELECT id, league FROM user WHERE score BETWEEN 2000 AND 3000;
```

#NULL SELECT
```
SELECT * FROM user WHERE score IS NULL;
SELECT * FROM user WHERE score IS NOT NULL;
```

#중복제거
```
SELECT DISTINCT league FROM user;
```

```
#오름 차순(ASC) / 내림 차순(DESC)
SELECT name,score FROM user ORDER BY name;
SELECT name,score FROM user ORDER BY name DESC;

#정렬과 중복제거는 검색성능을 느리게 만듭니다.

#특정 문자열 검색 (% = all, _ = 자릿수)
SELECT id,name FROM user WHERE id LIKE 'a%';
SELECT id,name FROM user WHERE id LIKE '_m';

#DELETE
DELETE FROM user WHERE id = 'user';

#UPDATE
UPDATE user SET league='S' WHERE id = 'id001';
UPDATE user SET score = score - 100 WHERE id = 'id001';

#ADD Column
ALTER TABLE user ADD COLUMN birth DATE;
ALTER TABLE user ADD COLUMN birth DATE FIRST;
ALTER TABLE user ADD COLUMN birth DATE AFTER name;

#DROP Column
ALTER TABLE user DROP COLUMN birth;

#CHANGE Column
ALTER TABLE user CHANGE COLUMN birth birthday DATE;

#ADD/DROP Constraint, Foreign Key
ALTER TABLE game ADD CONSTRAINT FOREIGN KEY (winner) REFERENCES user(id);
ALTER TABLE game DROP FOREIGN KEY constraint_name;
(show create table game - check constraint_name)

#외래키 or 참조키
CREATE TABLE user (
        id VARCHAR(12) NOT NULL PRIMARY KEY,
        password VARCHAR(12) NOT NULL,
        contact_id VARCHAR(12) NOT NULL,
        CONSTRAINT my_contact_contact_id_fk
        FOREIGN KEY (contact_id)
        REFERENCES my_contact (contact_id)
);

#CASE WHEN 1
SELECT
        CASE a
                WHEN 1 THEN 'true'
                WHEN 0 THEN 'false'
                ELSE 'what?'
        END
FROM user;
```

```
#CASE WHEN 2
SELECT
        CASE
                WHEN ratio > 0.9 THEN 'killer'
                WHEN ratio > 0.5 THEN 'human'
                ELSE 'what?'
        END
FROM user;

#CROSS JOIN
SELECT * FROM stu CROSS JOIN prof;
SELECT * FROM stu CROSS JOIN prof WHERE stu.pid = prof.id;

#INNER JOIN(Eqi-Join)
SELECT * FROM stu INNER JOIN prof ON stu.pid = prof.id;

#INNER JOIN(Theta-Join)
SELECT * FROM stu INNER JOIN prof ON stu.pid <> prof.id;

#UNION
(SELECT name FROM stu) UNION (SELECT name FROM prof);

#LEFT OUTER JOIN
SELECT * FROM stu S LEFT JOIN prof P ON S.pid = P.id;

#RIGHT OUTER JOIN
SELECT * FROM stu S RIGHT JOIN prof P ON S.pid = P.id;

#FULL OUTER JOIN
(SELECT * FROM stu S LEFT JOIN prof P ON S.pid = P.id)
UNION
(SELECT * FROM stu S RIGHT JOIN prof P ON S.pid = P.id);

#Sub Query
SELECT * FROM stu WHERE profid = (SELECT pid FROM prof WHERE pname LIKE 'lee%');
SELECT * FROM dojang WHERE id IN (3,5,6);
SELECT * FROM dojang WHERE id NOT IN (3,4);

# View
CREATE VIEW scoreboard AS SELECT person.name, sum(result.point)
FROM person JOIN result ON person.id = result.id GROUP BY result.id;

#Transaction Commit
START TRANSACTION;
…
COMMIT;

#Transaction Rollback
START TRANSACTION;
…
ROLLBACK;
```

#스칼라 집합에 사용가능한 명령어
IN - 결과 값중 하나와 일치하면 참
NOT IN - IN이 거짓이면 참
ANY(=SOME) - 결과값에 하나라도 만족하면 참
ALL - 결과값 전체가 조건을 만족해야 참

#집단함수
SELECT *league*, COUNT(*id*) FROM *user* WHERE *league*='b';
해당조건에 맞는 레코드를 카운드 합니다.
SELECT *league*, AVG(*score*) FROM *user* WHERE *league*='g';
해당조건에 맞는 레코드에서 지정한 필드의 평균을 구합니다.
SELECT *league*, AVG(*score*) FROM *user* GROUP BY *league*;
해당조건에 맞는 레코드를 지정한 필드에 값에 따라서 그룹화해서 정렬한다.
SELECT *league*, AVG(*score*) FROM *user* GROUP BY *league* HAVING AVG(*score*) >1000;
그룹화되 자료중 조건에 맞는 자료를 정렬한다.(where절은 앞에)
SELECT SUM(*num_play*)*10 AS *income* FROM *user*;
해당 자료의 스키마의 속성값을 AS뒤에 값으로 대치한다.

#내장함수
해당 속성이나 조건에 맞는 자료의 합(수식 계산가능)
SELECT SUM(*num_play*)*10 AS *income* FROM *user*;
현재시간
SELECT NOW() AS TIME;
해당 값의 로그값
SELECT LOG((10 + 20) * 30);
해당 값의 제곱 값(ex 2의 10승)
SELECT POW(2,10);
주어진 문자열을 합한다.(ex hello world)
SELECT CONCAT('hello','world');
UPDATE test SET b = CONCAT(b,'님');
주어진 문자열을 지정한 문자열로 대치한다.(hello -> hellow);
SELECT REPLACE('hello', 'o', 'ow');
UPDATE test SET b = REPLACE (b, '님', '씨');

#데이터 타입
Numeric
BIT (m) - 1~64
TINYINT (m) - 0~255 or -128~127
BOOL - true or false, 0 or 1
SAMLLINT (m) - 0~65535 or -32768~32767
MEDIUMINT (m) - 0~16777215 or -8388608~8388607
INT=INTEGER (m) - 0~4294967295 or -2147483648~2147483647
BIGINT - 0~18446744073709551615 or -9223372036854775808~9223372036854775807.
DEC=DECIMAL (m,d) - m = 전체 자릿수 (max 65), d = 소수점 자리수(max 30) / 10진수로 저장 / 저장 용량이 크다.
FLOAT (m,d) - 1.175494351E-38 ~ 3.402823466E+38 and -3.402823466E+38~-1.175494351E-38, 0
DOUBLE (m,d) - 2.225073858507201 4E-308 ~ 1.7976931348623157E+308 and -1.7976931348623157E+308 to-2.2250738585072014E-308

Date and Time
DATE - 'YYYY-MM-DD'
DATETIME (fsp) - 'YYYY-MM-DD HH:MM:SS[.fraction]'
TIMESTAMP (fsp) - 'YYYY-MM-DD HH:MM:SS[.fraction]' / 아무것도 입력되지 않은경우 테이블이 실행되거나 값이 입력되는 순간의 시간을 기록
TIME (fsp) - 'HH:MM:SS[.fraction]'
YEAR (2 ro 4) - YY or YYYY

String
CHAR (m) - 255로 제한 되었으나 현재는 없은 단, 레코드 크기에 의해서 조정, 일정한 글자의 유리
VARCHAR (m) - 입력되는 내용에 크기에 따라 크기변화 최대를 m으로 지정
BINARY / VARBINARY / TINYBLOB / TINYTEXT / TEXT / MEDIUMBLOB / MEDIUMTEXT / LONGBLOB / LONGTEXT
BLOB - 이미지 동영상도 저장 가능

Space - POINT / LINESTRING / POLYGON

#제약조건
PRIMARY KEY / NOT NULL / AUTO_INCREMENT / DEFAULT / UNIQUE / FOREIGN KEY

#한글 charset 문제
메모장이나 노트패드에서 charset 변경 및 저장