머신러닝 파이썬 활용 빅데이터 분석 개발자 양성 과정

MySQL 강의자료

2020.10

1. MySQL 설치

■ MySQL 설치

MySQL Community version (MySQL Installer for Windows 8.0.21)
중간에 Execute, Next
잘못 설치되면 C:\ProgramData\MySQL 이하 제거 및
regedit Registry 정리

■ 한글 사용(5.7 이하 버전에서는 필요함)

C:\ProgramData\MySQL\MySQL Server 5.7\my.ini 맨 마지막에 추가

```
[client]
default-character-set=utf8

[mysql]
default-character-set=utf8

[mysqld]
character-set-client-handshake = FALSE
init_connect="SET collation_connection = utf8_general_ci"
init_connect="SET NAMES utf8"
character-set-server = utf8

[mysqldump]
default-character-set = utf8
```

1. MySQL 설치

■ 한글 확인

```
show variables like ‘char%’;
ALTER DATABASE [DB명] DEFAULT CHARACTER SET utf8
alter database mysql default character set utf8;
```

■ Heidisql 설치 (https://www.heidisql.com/download.php)

Installer, 32/64 bit combined
신규, Root 폴더에 세션 생성
도구>환경설정: 기본, SQL, 격자서식설정 메뉴탭에서 폰트와 글자크기 변경

■ 사용자 생성 및 권한 부여

```
create user 'userID'@'%' identified by 'userpassword';
create user 'hsuser'@'localhost' identified by 'hspass';
grant all privileges on *.* to 'hsuser'@'localhost';
flush privileges;
```

■ 작업용 database 생성

```
create database test
  default character set utf8
  default collate utf8_general_ci;
```

2. 데이터 타입

■ 사용 가능한 데이터 타입

- Numerical
- Date and Time
- String
- Spatial
- JSON

2. 데이터 타입

■ Numerical Data Type

- 정수

타입	저장공간	최소값 (Signed)	최소값 (Unsigned)	최대값 (Signed)	최대값 (Unsigned)
TINYINT	1	-128	0	127	255
SMALLINT	2	-32768	0	32767	65535
MEDIUMINT	3	-2^{23}	0	$2^{23} - 1$	$2^{24} - 1$
INT	4	-2^{31}	0	$2^{31} - 1$	$2^{32} - 1$
BIGINT	8	-2^{63}	0	$2^{63} - 1$	$2^{64} - 1$

- 실수(정확한 값): NUMERIC, DECIMAL

```
salary DECIMAL(5, 2)
```

- 실수(근사치): FLOAT, DOUBLE

2. 데이터 타입

■ Date and Time Data Type

- DATE type: YYYY-MM-DD
- DATETIME type: YYYY-MM-DD hh:mm:ss

```
1000-01-01 00:00:00 ~ 9999-12-31 23:59:59
```

- TIMESTAMP type

```
1970-01-01 00:00:01 UTC ~ 2038-01-19 03:14:07 UTC
```

- Time Zone

```
2020-01-01 10:10:10-08:00
```

- Default value

```
dt DATETIME DEFAULT CURRENT_TIMESTAMP
```

2. 데이터 타입

■ String Data Type

- CHAR and VARCHAR

Value	CHAR(4)	저장 공간	VARCHAR(4)	저장 공간
‘ ‘	‘ ‘	4	‘ ‘	1
‘ab’	‘ab ‘	4	‘ab’	3
‘abcd’	‘abcd’	4	‘abcd’	5
‘abcdefgh’	‘abcd’	4	‘abcd’	5

- BLOB(Binary Large OBject)

3. 테이블 조작

■ 테이블 생성

```
use test;
create table if not exists address_book (
    no int unsigned not null auto_increment,
    name varchar(10) not null,
    tel varchar(14),
    nickname varchar(20) default ‘별명’,
    primary key(no)
) auto_increment=10001;

create Table: CREATE TABLE `wp_options` (
    `option_id` bigint(20) unsigned NOT NULL AUTO_INCREMENT,
    `option_name` varchar(64) NOT NULL DEFAULT '',
    `option_value` longtext NOT NULL,
    `autoload` varchar(20) NOT NULL DEFAULT 'yes',
    PRIMARY KEY (`option_id`),
    UNIQUE KEY `option_name` (`option_name`)
) ENGINE=MyISAM AUTO_INCREMENT=1203 DEFAULT CHARSET=utf8

show table status;
```

3. 테이블 조작

■ 테이블 조회

```
show tables;
desc address_book;
```

■ 테이블 제거

```
drop table [테이블 명];
create table tmp (
    id int,
    name varchar(10)
);
drop table tmp;
```

■ 테이블 이름 변경

```
rename table [테이블 명] to [새 테이블명];
rename table tmp to tmp_table;
```

3. 테이블 조작

■ 테이블 변경

alter table

ADD	컬럼 추가
DROP	컬럼 삭제
CHANGE	컬럼명 변경, 컬럼 자료형 변경
MODIFY	컬럼 순서 바꾸기

▲ 컬럼 추가

```
alter table [테이블 명] add [컬럼명] 자료형;    # 맨 뒤에 추가
alter table address_table add gender char(2) not null;  # 남/여

alter table [테이블 명] add [컬럼명] 자료형 first; # 맨 앞에 추가
alter table [테이블 명] add [컬럼명] 자료형 after [앞 컬럼명];
/* 지정 컬럼 뒤에 추가 */
```

▲ 컬럼 삭제

```
alter table [테이블 명] drop [컬럼명];
```

3. 테이블 조작

■ 테이블 변경

alter table

ADD	컬럼 추가
DROP	컬럼 삭제
CHANGE	컬럼명 변경, 컬럼 자료형 변경
MODIFY	컬럼 순서 바꾸기

▲ 컬럼명 변경, 컬럼 자료형 변경

```
alter table [테이블 명] change [기존 컬럼명] [새 컬럼명] 자료형;
alter table [테이블 명] change [컬럼명] [컬럼명] 새 자료형;
alter table address_book change no aid int(8);
alter table address_book
                change aid aid int(4) unsigned auto_increment;
```

▲ 컬럼 순서 바꾸기

```
alter table [테이블 명] modify [컬럼명] 자료형 first;
alter table [테이블 명] modify [컬럼명] 자료형 after [다른 컬럼명];
alter table address_book modify gender char(2) not null after name;
```

4. 데이터 조작(SELECT)

■ 데이터 조회 - 조건

```
use world;
show tables;
desc city;

select * from city;

select * from city where countrycode='kor';
select * from city where district='kyonggi';
select name, population from city where district='kyonggi';
select * from city where district='kyonggi' and population>500000;
select name, population from city
    where district='kyonggi' and population>500000;
select district from city where countrycode='kor';
select distinct district from city where countrycode='kor';
select * from mysql.address_book;
select * from city where district='chollanam' or
    district='chollabuk' or district='kwangju';
```

4. 데이터 조작(SELECT)

■ 데이터 조회 – 조건 및 순서

```
select * from city where
    countrycode='kor' and population>1000000 and population%2=0;
select * from city where
    countrycode='kor' and population between 1000000 and 2000000;
select * from city where countrycode='kor‘ and name like 'tae%';

select * from city where district='kyonggi' order by name;
select * from city where district='kyonggi' order by name desc;
select * from city where district='kyonggi' order by population desc;
select * from city where countrycode='kor' order by district;
select * from city
    where countrycode='kor' order by district, population;
select * from city
    where countrycode='kor' order by district desc, population;
select * from city
    where countrycode='kor' order by district desc, population desc;
```

■ 데이터 조회 – 함수 이용, 그룹핑

```
select count(*) from city where countrycode='kor';
select sum(population) from city where countrycode='kor';
select avg(population) from city where countrycode='kor';
select max(population) from city;
select min(population) from city;
select min(population), max(population), avg(population),
    sum(population) from city where countrycode = 'kor';

select group_concat(name) from city where district='chungchongnam';
select group_concat(distinct district) from city
    where countrycode='kor';
select district, count(*) from city where countrycode='kor';
select * from city where countrycode='kor' group by district;
select district, count(*) from city
    where countrycode='kor' group by district;
select district, count(*) from city where countrycode='kor'
    group by district having count(*)=6;
select district, count(*) from city where countrycode='kor'
    group by district having count(*)>=6 order by count(*) desc;
```

4. 데이터 조작(SELECT)

■ 데이터 조회 – 그룹핑, 조인 예

```
select countrycode, count(*) from city
    group by countrycode having count(*) >=50;
select countrycode, count(*) from city group by countrycode
    having count(*) >=50 order by count(*) desc;

desc country;
desc countrylanguage;

select city.Name, city.Population, country.Name from city
    inner join country on city.CountryCode = country.Code;
select city.Name, city.Population, country.Name from city
    inner join country on city.CountryCode = country.Code
    where city.CountryCode='kor';
select city.Name, city.Population, country.Name from city
    inner join country on city.CountryCode = country.Code
    where city.Population > 7000000;
```

4. 데이터 조작(UPDATE & INSERT)

■ 데이터 갱신

```
use world;
update city set name='Siheung'
    where countrycode='kor' and name='Shihung';
update city set name='Siheung', population=153443
    where countrycode='kor' and name='Shihung';
```

■ 데이터 삽입

```
insert into city values (default, 'Gimpo', 'KOR', 'Kyonggi', 200001);
insert into city (name, countrycode, district, population)
    values ('Hwasong', 'KOR', 'Kyonggi', 312345);

insert into city (name, countrycode, district, population)
    values ('Osan', 'KOR', 'Kyonggi', 201234),
    ('Pochon', 'KOR', 'Kyonggi', 156789);
select * from city order by id desc limit 3;
```

4. 데이터 조작(UPDATE)

■ 데이터 삽입

```
create table citycopy like city;
show tables;
select * from citycopy;
insert into citycopy select * from city;
select * from citycopy;
```

■ 데이터 삭제

```
delete from city where id=4082;
select * from city order by id desc limit 3;
```

■ 뷰 생성

```
create view LargeCity as select * from city
    where population>7000000 with check option;
select * from largecity;
show tables;
```

4. 데이터 조작(Sub Query & Join)

■ Sub Query

```
create view KoreanCity as select id, name, district, population
    from city where countrycode='kor';
select * from KoreanCity ;
select district, name, population from koreancity as c1
    where population > (select avg(population) from koreancity as c2
    where c1.district = c2.district group by district);
```

■ Join

▲ Inner Join

```
select countrylanguage.`*`, country.name from countrylanguage
  inner join country on countrylanguage.countrycode=country.Code
  where language='korean';
```

▲ Outer Join

```
select city.`*`, country.Name from city left outer join country
  on city.CountryCode=country.Code where city.CountryCode='kor';
```

4. 데이터 조작(날짜 형식)

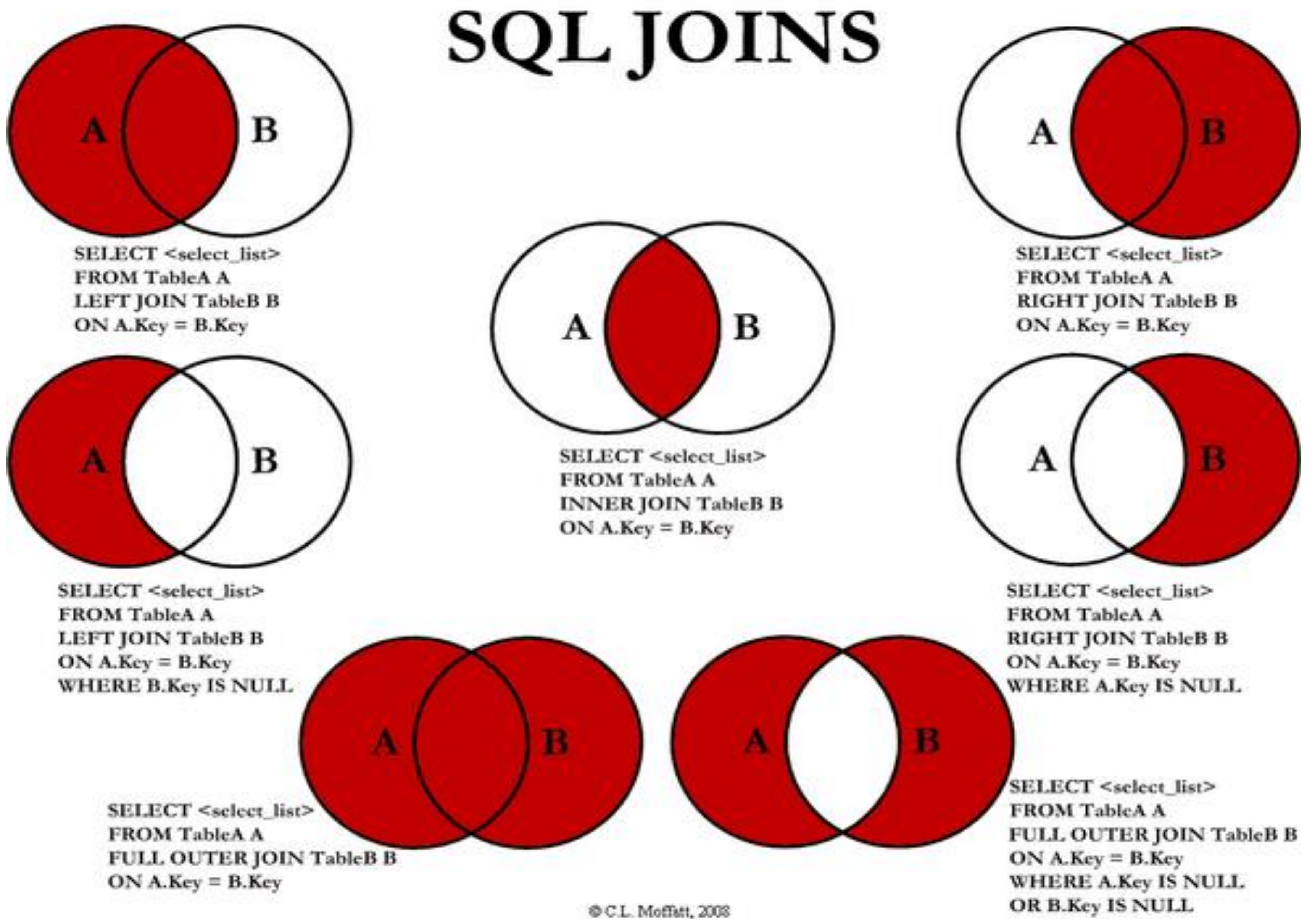
```
CREATE TABLE date_table (
    id int auto_increment primary key,
    datetime datetime default now()
) auto_increment=101;

INSERT INTO date_table (datetime) VALUES
    ('2017-08-28 17:22:21'), ('2017-02-15 10:22:24'),
    ('2017-12-09 22:13:24'), ('2018-07-06 20:15:18');
INSERT INTO date_table VALUES (default, default);

SELECT datetime FROM date_table where id=101;
SELECT date_format(datetime, '%Y-%m-%d') FROM date_table where id=101;
SELECT date_format(datetime, '%p %h:%i:%s')     /* %r */
    FROM date_table where id=103;

SELECT now(), curdate(), curtime();
SELECT date_add(now(), interval 2 MONTH);
SELECT date_sub(now(), interval 5 DAY);
SELECT to_days('2019-11-14') - to_days(now());  /* from AD */
SELECT dayofweek(datetime) FROM date_table;     /* 일요일: 1 */
```

5. 테이블 조인(Join)



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

5. 테이블 조인(Join)

■ 테이블 생성

```
CREATE TABLE girl_group (
    _id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(32) NOT NULL,
    debut DATE NOT NULL,
    hit_song_id INT
) DEFAULT CHARSET=utf8;

CREATE TABLE song (
    _id INT PRIMARY KEY AUTO_INCREMENT,
    title VARCHAR(32) NOT NULL,
    lyrics VARCHAR(32)
) DEFAULT CHARSET=utf8;
```

5. 테이블 조인(Join)

■ 데이터 삽입

```
INSERT INTO song VALUES (101, 'Tell Me', 'tell me tell me tetetete tel me');
INSERT INTO song (title, lyrics)
    VALUES ('Gee', 'GEE GEE GEE GEE GEE BABY BABY'),
    ('미스터', '이름이 뭐야 미스터'),
    ('Abracadabra', '이러다 미쳐 내가 여리여리'),
    ('8282', 'Give me a call Baby baby'), ('기대해', '기대해'),
    ('I Don\'t car', '다른 여자들의 다리를'),
    ('Bad Girl Good Girl', '앞에선 한 마디 말도'), ('피노키오', '뉴예삐오'),
    ('별빛달빛', '너는 내 별빛 내 마음의 별빛'),
    ('A', 'A 워오우 워오우워 우우우'),
    ('나혼자', '나 혼자 밥을 먹고 나 혼자 영화 보고'), ('LUV', '설레이나요 '),
    ('짧은치마', '짧은 치마를 입고 내가 길을 걸으면'),
    ('위아래', '위 아래 위위 아래'), ('Dumb Dumb' , '너 땜에 하루종일');

INSERT INTO girl_group (name, debut, hit_song_id)
    VALUES ('원더걸스', '2007-02-10', 101),
    ('소녀시대', '2007-08-02', 102), ('카라', '2009-07-30', 103),
    ('브라운아이드걸스', '2008-01-17', 104), ('다비치', '2009-02-27', 105),
    ('2NE1', '2009-07-08', 107), ('f(x)', '2011-04-20', 109),
    ('시크릿', '2011-01-06', 110), ('레인보우', '2010-08-12', 111);
INSERT INTO girl_group (name, debut)
    ('애프터 스쿨', '2009-11-25'), ('포미닛', '2009-08-28');
```

5. 테이블 조인(Join)

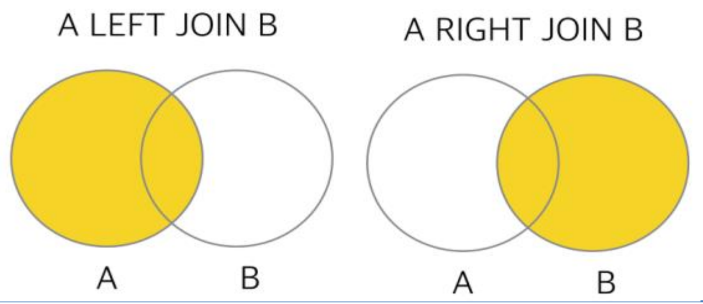
■ Inner Join

```
SELECT gg._id, gg.name, s.title
    FROM girl_group AS gg
    INNER JOIN song AS s              # JOIN song AS s
    ON s._id = gg.hit_song_id;        # WHERE s._id = gg.hit_song_id;
```

■ Left Outer, Right Outer Join

```
SELECT gg._id, gg.name, s.title
    FROM girl_group AS gg
    LEFT OUTER JOIN song AS s         # LEFT JOIN song AS s
    ON s._id = gg.hit_song_id;

SELECT s._id, s.title, gg.name
    FROM girl_group AS gg
    RIGHT OUTER JOIN song AS s        # RIGHT JOIN song AS s
    ON s._id = gg.hit_song_id;
```



5. 테이블 조인(Join)

■ 연습 문제

1) 2009년도에 데뷔한 걸그룹 정보를 조회
   (where debut between ‘2009-01-01’ and ‘2009-12-31’ 이용)

2) 2009년도에 데뷔한 걸그룹의 히트송은?
   (걸그룹 이름, 데뷔일, 히트송)

3) 대륙별로 국가숫자, GNP의 합, 평균 국가별 GNP는?

4) 아시아 대륙에서 인구가 가장 많은 도시 10개를 내림차순으로 보여줄 것
   (대륙명, 국가명, 도시명, 인구수)

5) 전 세계에서 인구가 가장 많은 10개 도시에서 사용하는 공식언어는?
   (도시명, 인구수, 언어명)

■ 테이블 Export/Import (in HeidiSQL)

```
SHOW VARIABLES LIKE "secure_file_priv";
    ➔ C:/ProgramData/MySQL/MySQL Server 5.7/Uploads

SELECT * FROM song INTO OUTFILE
    ‘C:/ProgramData/MySQL/MySQL Server 5.7/Uploads/song.csv'
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

TRUNCATE song;

LOAD DATA INFILE
    'c:/ProgramData/MySQL/MySQL Server 5.7/Uploads/song.csv'
    INTO TABLE song
    FIELDS TERMINATED BY ','
    ENCLOSED BY '"'
    LINES TERMINATED BY '\n';

SELECT * FROM song;
```

6. 기타

■ Key의 종류

- 후보키 (Candidate Key) :
  - 테이블을 구성하는 열 중에서 유일하게 식별할 수 있는 열
- 기본키 (Primary Key) :
  - 테이블에서 유일하게 식별하기 위해 사용하는 키
- 대체키 (Alternate Key) :
  - 후보키 중 기본키를 제외한 나머지 후보키
- 외래키 (Foreign Key) :
  - 테이블 내의 열 중 다른 테이블의 기본키를 참조하는 열
- 슈퍼키 (Super Key) :
  - 슈퍼키 또는 합성키라 불림
  - 2개 이상의 열이 합쳐서 기본키로 사용하는 것

7. NodeJS 에서 MySQL 사용법

- **프로그램 설치**

```
npm install mysql
```

- **샘플 프로그램**

```
const mysql = require('mysql');
let connection = mysql.createConnection({
    host     : 'localhost',
    user     : ‘hsuser',
    password : ‘hspass',
    database : 'world'
});

connection.connect();

let sql = 'SELECT * FROM city WHERE population > 9000000;';
connection.query(sql, function (error, results, fields) {
    if (error)
        throw error;
    console.log(results);
});

connection.end();
```

7. NodeJS 에서 MySQL 사용법

▪ **Config file을 이용한 샘플 프로그램**

```
const fs = require('fs');
const mysql = require('mysql');
const data = fs.readFileSync('./dbconfig.json');
const conf = JSON.parse(data);

const connection = mysql.createConnection({
         host: conf.host,
         user: conf.user,
         password: conf.password,
         port: conf.port,
         database: conf.database
});
connection.connect();

let sql = 'SELECT * FROM city WHERE population > 9000000;';
connection.query(sql, function (error, results, fields) {
    if (error) throw error;
    for (let city of results)
        console.log(city.Name, city.CountryCode, city.Population);
});

connection.end();
```

7. NodeJS 에서 MySQL 사용법

- **Insert 샘플 프로그램**

```
connection.connect();

let image = "https://placeimg.com/100/100/any";
let name = "유강남";
let birthday = "911004";
let gender = "남자";
let job = "야구선수";
let params = [image, name, birthday, gender, job];

const sql = 'insert into customer(image, name, birthday, gender, job) \
             values (?, ?, ?, ?, ?)';
connection.query(sql, params, function (error, results) {
    if (error)
        throw error;
});

connection.end();
```

7. NodeJS 에서 MySQL 사용법

```
getDB(params, callback) {
    connection.connect();
    let sql = 'SELECT * FROM ...;';
    connection.query(sql, params, function (error, results, fields) {
        if (error)
           throw error;
        callback(results);
    });
    connection.end();
}

writeDB(params, callback) {
    connection.connect();
    let sql = ‘INSERT INTO ...;';
    connection.query(sql, params, function (error, fields) {
        if (error)
           throw error;
        callback();
    });
    connection.end();
}
```