Mysql 설치 및 환경 설정(on ubuntu linux)

1. SQL(Structured Query Language)

- 관계형 데이터베이스 관리 시스템에서 데이터를 관리하기 위해 사용되는 표준 프로그래밍 언어(Language)
- 데이터베이스마다 문법에 약간의 차이가 있지만, 표준 SQL을 따름.
- SQL은 크게 세 가지 종류로 나뉨
  - 데이터 정의 언어(DDL, Data Definition Language)
  - 데이터 처리 언어(DML, Data Manipulation Language)
  - 데이터 제어 언어(DCL, Data Control Language)

1.1 데이터 정의 언어(DDL, Data Definition Language)

: 테이블의 구조 정의

- 테이블(TABLE), 인덱스(INDEX) 등의 개체를 만들고 관리하는데 사용되는 명령
- CREATE, ALTER, DROP 등이 있음

1.2 데이터 조작 언어(DML, Data Manipulation Language)

: 데이터 CRUD [Create(생성), Read(읽기), Update(갱신), Delete(삭제)]

- INSERT 테이블(Table)에 하나 이상의 데이터 추가.
- UPDATE 테이블(Table)에 저장된 하나 이상의 데이터 수정.
- DELETE 테이블(Table)의 데이터 삭제.
- SELECT 테이블(Table)에 저장된 데이터 조회.

1.3 데이터 제어 언어(DCL, Data Control Language)

: DB 핸들링 권한 설정, 데이터 무결성 처리 등 수행

- GRANT 데이터베이스 개체(테이블, 인덱스 등)에 대한 사용 권한 설정.
- BEGIN 트랜잭션(Transaction) 시작.
- COMMIT 트랜잭션(Transaction) 내의 실행 결과 적용.
- ROLLBACK 트랜잭션(Transaction)의 실행 취소.

2. MySQL 이해 및 실습

- MySQL는 오라클사가 소유.
- 불확실한 라이선스 정책으로 인해, 동일 소스를 기반으로 MariaDB 가 파생됨
- 거의 유사, MariaDB가 오픈소스 라이센스를 따르고 있다고 보면 됨

2.1 MySQL 설치 - 실습1

1. 리눅스 update 실행 $sudoapt-getupdate2.mysql-server$설치 sudo apt-get install mysql-server

3. mysql root 계정 비밀번호 및 secure 설정 $ sudo mysql_secure_installation

```
ubuntu@ip-172-31-9-225:~$ sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.
The 'validate_password' plugin is installed on the server.
The subsequent steps will run with the existing configuration
of the plugin.
Please set the password for root here.

New password:

Re-enter new password:

Estimated strength of the password: 50
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
ubuntu@ip-172-31-9-225:~$
```

4. sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
- 다음 설정을 [mysqld] 에 추가
1. mysql 원격 접속 허용
- bind-address 0.0.0.0 로 변경
2. mysql 한글 설정 추가
collation-server = utf8_unicode_ci
character-set-server = utf8
skip-character-set-client-handshake
5. mysql 서비스 시작
$ sudo service mysql start

[참고]

Usage: /etc/init.d/mysql start|stop|restart|status

$ sudo service mysql restart

[참고]

/etc/mysql/mysql.conf.d/mysqld.cnf 파일의 [mysqld] 설정 예

```
[mysqld]
\#############################
\#  Basic Settings
\#
user            = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port            = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
lc-messages-dir = /usr/share/mysql

skip-external-locking
collation-server = utf8_unicode_ci
character-set-server = utf8
skip-character-set-client-handshake

\#
\# Instead of skip-networking the default is now to listen only on
\# localhost which is more compatible and is not less secure.

bind-address    = 0.0.0.0
#############################
```

참고1: mysql DB 설정 값을 저장하고 있는 DB

mysql> use mysql;

- use 데이터베이스명;
- mysql 데이터베이스는 mysql 설정을 저장하고 있는 데이터베이스임.
- 해당 데이터베이스에서 작업을 하겠다는 의미임.

참고2: mysql 접속 허용 관련 설정

1) 로컬에서만 접속 허용
mysql> GRANT ALL PRIVILEGES ON DATABASE.TABLE to 'root'@localhost identified by "korea1234";

2) 특정 호스트에만 접속 허용
mysql> GRANT ALL PRIVILEGES ON DATABASE.TABLE to 'root'@www.blim.co.kr identified by "korea1234";

3) 모든 호스트에서 접속 허용
mysql> GRANT ALL PRIVILEGES ON DATABASE.TABLE to 'root'@'%' identified by "korea1234";

[옵션 상세 설명]

(1) ALL – 모든 권한 / SELECT, UPDATE – 조회, 수정 권한등으로 권한 제한 가능
예) GRANT INSERT,UPDATE,SELECT ON . TO 'username'@'localhost' IDENTIFIED BY '비밀번호';

(2) database.table
– 특정 데이터베이스에 특정 테이블에만 권한을 줄 수 있음 / .
– 모든 데이터베이스에 모든 테이블 권한을 가짐

(3) root – 계정명

(4) korea1234 – 계정 비밀번호

참고3: flush privileges;

```
- user, db 같은 grant table을 INSERT, UPDATE 등을 써서 직접 데이터 입력/수정을 할 경우,
  grant tables를 다시 읽어야 권한 설정이 적용됨.
- 서버를 재기동하지 않고, grant table을 새로 읽으라는 명령이 flush privileges 이지만,
  INSERT, UPDATE 가 아닌 GRANT 명령을 사용했을 경우에는 해당 명령 생략 가능
  (하지만 확실하게 하기 위해 일반적으로 사용)
```

mysql 접속 명령

$sudomysql-hhost-uuser-p[password]dbnamehost:mysql$서버주소(관련옵션사용안하면, 현재$PC)user:mysqlIDdpassword:mysqlID$의암호예)
sudo mysql -u root -p 현재 PC에 root ID로 별도 데이터베이스를 지정하지 않고, 접속

실습3 - 데이터베이스, 테이블 목록 보기

```
$ sudo mysql -u root -p
mysql> show databases;
mysql> use mysql;
mysql> show tables;
mysql> use information_schema;
mysql> show tables;
mysql> exit

참고: https://dev.mysql.com/doc/refman/5.7/en/
```

mysql 데이터베이스 서버 실행 / 종료

```
- mysql 실행
$ sudo service mysql start
- mysql 재실행
$ sudo service mysql restart
- mysql 멈춤
$ sudo service mysql stop
- 서비스 상태 확인
$ service mysql status
```

참고: https://dev.mysql.com/doc/refman/5.7/en/

3. SQL DDL(Data Definition Language) 이해 및 실습

3.1 데이터베이스

- 데이터베이스 안에는 여러 개의 데이터베이스 이름이 존재한다.
- 각 데이터베이스 이름 안에는 여러 개의 테이블이 존재한다.

```
1. 데이터베이스 생성
mysql> CREATE DATABASE mydb;

2. 데이터베이스 목록 보기
mysql> SHOW DATABASES;

3. dbname 데이터베이스 사용 시
mysql> USE mydb;

4. dbname 데이터베이스 삭제
mysql> DROP DATABASE [IF EXISTS] mydb;

IF EXISTS 옵션은 해당 데이타베이스 이름이 없더라도 오류를 발생시키지 말라는 의미
```

- CREATE SCHEMA mydb;
  CREATE DATABASE mydb과 동일한 명령임

In [ ]:

실습4 - 데이터베이스 생성, 목록 보기, 사용, 삭제

```
mysql> CREATE DATABASE sqltest;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| sqltest            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> USE sqltest;
Database changed
mysql> DROP DATABASE IF EXISTS sqltest;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)
```

3.2 테이블

3.2.1 테이블 생성

- 기본 문법 (CREATE TABLE 구문)
  ```
  CREATE TABLE 테이블명 (
    컬럼명 데이터형,
    컬러명 데이터형,
    .
    .
    기본키 셋
  );
  ```

1. 테이블을 생성할 데이터베이스를 먼저 사용하겠다고 명령한 후에,
   ```
   mysql> CREATE DATABASE mydb;
   mysql> use mydb;
   ```
2. 테이블 생성
   ```
   mysql> CREATE TABLE 테이블명 (
    컬럼명 데이터형,
    컬러명 데이터형,
    .
    .
    Primary Key 가 될 필드 지정
   );
   </div>
   ```

- 숫자 타입의 컬럼 정의 문법

```
mysql> CREATE TABLE mytable (
  -> id INT [UNSIGNED] [NOT NULL] [AUTO_INCREMENT],
id : 컬럼명, 가능한 영어 소문자 중심으로 명명
INT : 컬럼에 대한 데이터 타입 선언
[UNSIGNED] : 옵션 사항
           예) TINYINT 로 지정시 -128 ~ +127
               TINYINT UNSIGNED 로 지정시 0 ~ 255
[NOT NULL] : NOT NULL 명시하면 데이터 입력시, 해당 컬럼 데이터에 값이 할당되지 않는 경우를 허락하지 않겠다는 의미
[AUTO_INCREMENT] : AUTO_INCREMENT 명시하면, 해당 테이블에 데이터 등록시 해당 컬럼은 자동으로 숫자가 1씩 증가하여 저장됨
                   해당 컬럼은 양의 정수만 등록할 수 있어야 하고, 테이블 안에서 AUTO_INCREMENT 컬럼은 하나만 지정 가능함
```

숫자형 데이터 타입

데이터 유형	정의
TINYINT	정수형 데이터 타입(1byte) -128 ~ +127 또는 0 ~ 255수 표현 가능
SMALLINT	정수형 데이터 타입(2byte) -32768 ~ 32767 또는 0 ~ 65536수 표현 가능
MEDIUMINT	정수형 데이터 타입(3byte) -8388608 ~ +8388607 또는 0 ~ 16777215수 표현 가능
INT	**정수형 데이터 타입(4byte) -2147483648 ~ +2147483647 또는 0 ~ 4294967295수 표현 가능**
BIGINT	정수형 데이터 타입(8byte) - 무제한 수 표현 가능
FLOAT(정수부 길이, 소수부 자릿수)	부동 소수형 데이터 타입(4byte)
DECIMAL(정수부 길이, 소수부 자릿수)	고정 소수형 데이터 타입고정(길이+1byte) **예), DECIMAL(5, 2) → 12345.67 과 같은 수 표현**
DOUBLE(정수부 길이, 소수부 자릿수)	부동 소수형 데이터 타입(8byte)

- 문자 타입의 컬럼 정의 문법

```
mysql> CREATE TABLE mytable (
  -> name VARCHAR(50),
name : 컬럼명, 가능한 영어 소문자 중심으로 명명
VARCHAR(n) : 컬럼에 대한 문자형 데이터 타입 선언
```

문자형 데이터 타입

데이터 유형	정의
CHAR(n)	고정 길이 데이터 타입, 정확히 (n <= 255) 예) CHAR(5) 'Hello'는 5 byte 사용, CHAR(50) 'Hello'는 50 byte 사용
VARCHAR(n)	**가변 길이 데이터 타입 (n <= 65535)** 예) VARCHAR(50) 'Hello'는 5 byte만 사용
TINYTEXT(n)	문자열 데이터 (n <= 255)
TEXT(n)	문자열 데이터 (n <= 65535)
MEDIUMTEXT(n)	문자열 데이터 (n <= 16777215)
LONGTEXT(n)	문자열 데이터 (n <= 4294967295)

- 시간 타입의 컬럼 정의 문법

```
mysql> CREATE TABLE coz_table (
  -> ts DATE,
ts : 컬럼명, 가능한 영어 소문자 중심으로 명명
DATE : 컬럼에 대한 시간 타입 선언
```

시간형 데이터 타입

데이터 유형	정의
DATE	**날짜(YYYY-MM-DD) 형태의 기간 표현 데이터 타입(3byte)**
TIME	시간(hh:mm:ss) 형태의 기간 표현 데이터 타입(3byte)
DATETIME	날짜와 시간(YYYY-MM-DD hh:mm:ss) 형태 '1001-01-01 00:00:00' 부터 '9999-12-31 23:59:59' 까지의 값 표현
TIMESTAMP	1970-01-01 00:00:00 이후부터 시스템 현재 시간까지의 지난 시간을 초로 환산하여 숫자로 표현
YEAR	YEAR(n)과 같은 형식으로 사용 n은 2와 4 지정 가능 2인 경우는 70 에서 69 까지, 4인 경우는 1970 에서 2069 까지 표시

- Primary Key 가 될 필드 지정 문법

```
mysql> CREATE TABLE mytable (
  -> 컬럼명 데이터형,
  -> .
  -> PRIMARY KEY(컬럼명1, 컬럼명2, ...)
  -> );
컬럼명1, 컬럼명2, ... : PRIMARY KEY 로 지정할 컬럼명을 넣음 (한 개 이상을 지정할 수 있음, 보통은 한 개를 지정함)
       PRIMARY KEY 로 지정할 컬럼은 NULL 값을 등록할 수 없어야 하고, 컬럼 안에서 같은 값이 없도록 각 값이 유일해야 함
       따라서, 해당 컬럼은 보통 NOT NULL(NULL 값 방지) AUTO_INCREMENT(유일함) 선언이 되어 있는 경우가 많음
```

```
예)
CREATE TABLE mytable (
id INT UNSIGNED NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
modelnumber VARCHAR(15) NOT NULL,
series VARCHAR(30) NOT NULL,
PRIMARY KEY(id)
);
```

실습5 - 테이블 생성

1. 데이터베이스 만들기: 이름 customer_db
2. 테이블 만들기

테이블명	컬럼명	데이터형식	NULL 유무	기본키	외래키	FK 테이블명	FK 컬럼명
고객테이블	no	INTEGER	NOT NULL	PK			
(customer)	name	CHAR(20)	NOT NULL				
	age	TINYINT					
	phone	VARCHAR(20)					
	email	VARCHAR(30)	NOT NULL				
	address	VARCHAR(50)					

3. 테이블 잘못만들어졌으면 DROP TABLE [IF EXISTS] dbname; 로 테이블 삭제 후 재생성

```
CREATE DATABASE customer_db;
USE customer_db;

CREATE TABLE mytable (
    id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name CHAR(20) NOT NULL,
    age TINYINT,
    phone VARCHAR(20),
    email VARCHAR(30) NOT NULL,
    address VARCHAR(50),
    PRIMARY KEY(id)
);
```

3.2.1 테이블 조회

- 기본 문법 (SHOW TABLES)

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_cozlab |
+------------------+
| mytable          |
+------------------+
1 row in set (0.00 sec)
```

- 기본 문법 (DESC 테이블명)

```
mysql> desc mytable;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id          | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name        | varchar(50)      | NO   |     | NULL    |                |
| modelnumber | varchar(15)      | NO   |     | NULL    |                |
| series      | varchar(30)      | NO   |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

3.2.2 테이블 삭제

- 기본 문법 (DROP TABLE 테이블명)
  `mysql> DROP TABLE [IF EXISTS] 테이블명;`

IF EXISTS 옵션은 해당 데이타베이스 이름이 없더라도 오류를 발생시키지 말라는 의미

3.2.3 테이블 구조 수정

- 테이블에 새로운 컬럼 추가

```
문법: ALTER TABLE [테이블명] ADD COLUMN [추가할 컬럼명][추가할 컬럼 데이터형]
mysql> ALTER TABLE mytable ADD COLUMN model_type varchar(10) NOT NULL;
```

- 테이블 컬럼 타입 변경

```
문법: ALTER TABLE [테이블명] MODIFY COLUMN [변경할 컬럼명][변경할 컬럼 타입]
mysql> ALTER TABLE mytable MODIFY COLUMN name varchar(20) NOT NULL;
```

- 테이블 컬럼 이름 변경

문법: ALTER TABLE [테이블명] CHANGE COLUMN [기존 컬럼 명][변경할 컬럼 명][변경할 컬럼 타입]
mysql> ALTER TABLE mytable CHANGE COLUMN modelnumber model_num varchar(10) NOT NULL;

- 테이블 컬럼 삭제

문법: ALTER TABLE [테이블명] DROP COLUMN [삭제할 컬럼 명]
mysql> ALTER TABLE mytable DROP COLUMN series;

실습5 - 테이블 생성, 조회

```
mysql> CREATE DATABASE mydb;
mysql> USE mytable;
mysql> CREATE TABLE mytable (
    ->   id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    ->   name VARCHAR(50) NOT NULL,
    ->   modelnumber VARCHAR(15) NOT NULL,
    ->   series VARCHAR(30) NOT NULL,
    ->   PRIMARY KEY(id)
    -> );
mysql> SHOW TABLES;
+------------------+
| Tables_in_mydb   |
+------------------+
| mytable          |
+------------------+
1 row in set (0.00 sec)
mysql> desc mytable;
+-------------+------------------+------+-----+---------+----------------+
| Field       | Type             | Null | Key | Default | Extra          |
+-------------+------------------+------+-----+---------+----------------+
| id          | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name        | varchar(50)      | NO   |     | NULL    |                |
| modelnumber | varchar(15)      | NO   |     | NULL    |                |
| series      | varchar(30)      | NO   |     | NULL    |                |
+-------------+------------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

연습문제1: 다음과 같이 보이도록 테이블 컬럼을 수정하시오

```
mysql> desc mytable;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| id         | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name       | varchar(20)      | NO   |     | NULL    |                |
| model_num  | varchar(10)      | NO   |     | NULL    |                |
| model_type | varchar(10)      | NO   |     | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
```

연습문제2 - 위 테이블을 삭제한 후, 연습문제1과 같은 컬럼을 가진 테이블을 생성하시오.
(테이블명은 model_info 로 하시오)

```
CREATE TABLE mytable (
       id INT UNSIGNED NOT NULL AUTO_INCREMENT,
       name VARCHAR(50) NOT NULL,
       modelnumber VARCHAR(15) NOT NULL,
       series VARCHAR(30) NOT NULL,
       PRIMARY KEY(id)
);

ALTER TABLE mytable MODIFY COLUMN name varchar(20) NOT NULL;
ALTER TABLE mytable CHANGE COLUMN modelnumber model_num varchar(10) NOT NULL;
ALTER TABLE mytable CHANGE COLUMN series model_type varchar(10) NOT NULL;
```

4. SQL DML(Data Manipulation Language) 이해 및 실습

(focusing on CRUD)

4.1. CRUD [Create(생성), Read(읽기), Update(갱신), Delete(삭제)]

- 데이터 관리는 결국 데이터 생성, 읽기(검색), 수정(갱신), 삭제 를 한다는 의미

4.1.1 데이터 생성

- 테이블에 컬럼에 맞추어 데이터를 넣는 작업 - 기본 문법 (INSERT) 1. 테이블 전체 컬럼에 대응하는 값을 모두 넣기

```
    mysql> INSERT INTO 테이블명 VALUES(값1, 값2, ...);
```

2. 테이블 특정 컬럼에 대응하는 값만 넣기 (지정되지 않은 컬럼은 디폴트값 또는 NULL값이 들어감)

```
    mysql> INSERT INTO 테이블명 (col1, col2, ...) VALUES(값1, 값2, ...);
```

예)

```
$ sudo mysql -u root -p

mysql> show databases;
```

```
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mydb               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+

mysql> use mydb;
mysql> show tables;
+------------------+
| Tables_in_cozlab |
+------------------+
| mytable          |
+------------------+

mysql> desc mytable;
+------------+------------------+------+-----+---------+----------------+
| Field      | Type             | Null | Key | Default | Extra          |
+------------+------------------+------+-----+---------+----------------+
| id         | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name       | varchar(20)      | NO   |     | NULL    |                |
| model_num  | varchar(10)      | NO   |     | NULL    |                |
| model_type | varchar(10)      | NO   |     | NULL    |                |
+------------+------------------+------+-----+---------+----------------+
mysql> INSERT INTO mytable VALUES(1, 'i7', '7700', 'Kaby Lake');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM mytable;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
mysql> INSERT INTO mytable (name, model_num, model_type) VALUES('i7', '7700K', 'Kaby Lake');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM mytable;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
|  2 | i7   | 7700K     | Kaby Lake  |
+----+------+-----------+------------+
2 rows in set (0.00 sec)
```

실습 - 데이터 생성(입력), 검색

웹페이지(http://www.enuri.com/list.jsp?cate=070701) 에서 1위 ~ 5위까지 데이터 입력하기

- id는 자동 증가
- name 은 코어 이름
- model_num 은 코어 모델 번호
- model_type 은 코드명 (카비레이크 -> Kaby Lake, 스카이레이크 -> Sky Lake 로 영문으로 작성)

- 예: 코어 i7 7700 카비레이크
-> name 은 i7
-> model_num 은 7700
-> model_type 은 Kaby Lake

```
INSERT INTO mytable (name, model_num, model_type) VALUES('i7', '7700', 'Kaby Lake');
INSERT INTO mytable (name, model_num, model_type) VALUES('i7', '7500', 'Kaby Lake');
INSERT INTO mytable (name, model_num, model_type) VALUES('i7', '7700K', 'Kaby Lake');
INSERT INTO mytable (name, model_num, model_type) VALUES('i7', 'G4600', 'Kaby Lake');
INSERT INTO mytable (name, model_num, model_type) VALUES('i7', '7600', 'Kaby Lake');
```

4.1.2 데이터 읽기(검색)

- 테이블에 저장된 데이터를 읽는 작업
- 데이터베이스는 대용량 데이터를 가정하므로, 대용량 데이터 중 특정한 조건에 맞는 데이터를 추출하는 검색 작업이라고 하는 편이 보다 적합함
- 기본 문법 (SELECT)

1. 테이블 전체 컬럼의 데이터 모두 읽기

mysql> SELECT * FROM 테이블명;

2. 테이블 특정 컬럼의 데이터만 읽기

mysql> SELECT 컬럼1, 컬럼2, ... FROM 테이블명;

```
mysql> SELECT name, model_num FROM mytable;

+------+-----------+
| name | model_num |
+------+-----------+
| i7   | 7700      |
| i7   | 7700K     |
+------+-----------+
```

3. 테이블 특정 컬럼의 데이터를 검색하되, 표시할 컬럼명도 다르게 하기

mysql> SELECT 컬럼1 AS 바꿀컬럼이름, 컬럼2 AS 바꿀컬럼이름 FROM 테이블명;

예)

```
mysql> SELECT name AS cpu_name, model_num AS cpu_num FROM mytable;

+----------+---------+
| cpu_name | cpu_num |
+----------+---------+
| i7       | 7700    |
| i7       | 7700K   |
+----------+---------+
```

4. 데이터 정렬해서 읽기

- ORDER BY 정렬할 기준 컬럼명 DESC|ASC
- DESC는 내림차순 ASC는 오름차순

mysql> SELECT * FROM 테이블명 ORDER BY 정렬할기준컬럼명 DESC;

mysql> SELECT 컬럼1, 컬럼2 FROM 테이블명 ORDER BY 정렬할기준컬럼명 ASC;

예)

```
mysql> SELECT * FROM mytable ORDER BY id DESC;

+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  2 | i7   | 7700K     | Kaby Lake  |
|  1 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
2 rows in set (0.01 sec)
mysql> SELECT * FROM mytable ORDER BY id ASC;

+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
|  2 | i7   | 7700K     | Kaby Lake  |
+----+------+-----------+------------+
2 rows in set (0.00 sec)
```

5. 조건에 맞는 데이터만 검색하기 (비교)

- WHERE 조건문 으로 조건 검색
- 예) WHERE 컬럼명 < 값
- 예) WHERE 컬럼명 > 값
- 예) WHERE 컬럼명 = 값

SELECT * FROM 테이블명 WHERE 필드명 = '값'

예)

```
mysql> SELECT * FROM mytable WHERE id < 2;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
mysql> SELECT * FROM mytable WHERE id = 1;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
mysql> SELECT * FROM mytable WHERE id > 1;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  2 | i7   | 7700K     | Kaby Lake  |
+----+------+-----------+------------+
```

6. 조건에 맞는 데이터만 검색하기 (논리 연산자)

- WHERE 조건문 으로 조건 검색
- 논리 연산자 활용
- 예) WHERE 컬럼명 < 값 OR 컬럼명 > 값
- 예) WHERE 컬럼명 > 값 AND 컬럼명 < 값

SELECT * FROM 테이블명 WHERE (필드명='값') OR (필드명 ='값');

SELECT * FROM 테이블명 WHERE (필드명='값') AND (필드명 ='값');

예)

```
mysql> SELECT * FROM mytable WHERE id > 0 OR id < 2;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
```

```
|  1 | i7   | 7700      | Kaby Lake  |
|  2 | i7   | 7700K     | Kaby Lake  |
+----+------+-----------+------------+
2 rows in set (0.00 sec)

mysql> SELECT * FROM mytable WHERE id = 1 AND name = 'i7';
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
```

7. 조건에 맞는 데이터만 검색하기 (LIKE 를 활용한 부분 일치)

- WHERE 조건문 으로 조건 검색
- LIKE 활용
- 예) 홍으로 시작되는 값을 모두 찾을 경우

```
SELECT * FROM 테이블명 WHERE 필드명 LIKE '홍%';
```

- 예) 홍이 들어간 값을 모두 찾을 경우

```
SELECT * FROM 테이블명 WHERE 필드명 LIKE '%홍%';
```

- 예) 홍으로 시작되고 뒤에 2글자가 붙을 경우

```
SELECT * FROM 테이블명 WHERE 필드명 LIKE '홍__';
```

예)

```
mysql> SELECT * FROM mytable WHERE name LIKE 'i%';

+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
|  2 | i7   | 7700K     | Kaby Lake  |
+----+------+-----------+------------+
2 rows in set (0.00 sec)
mysql> SELECT * FROM mytable WHERE name LIKE 'i__';
Empty set (0.00 sec)
```

8. 결과중 일부만 데이터 가져오기 (LIMIT 을 활용)

- LIMIT 활용
- 예) 결과중 처음부터 10개만 가져오기

```
SELECT * FROM 필드명 LIMIT 10;
```

- 예) 결과중 100번째부터, 10개만 가져오기

```
SELECT * FROM 필드명 LIMIT 100, 10;
```

예)

```
mysql> SELECT * FROM mytable LIMIT 1;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  1 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
mysql> SELECT * FROM mytable LIMIT 1, 1;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  2 | i7   | 7700K     | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
```

9. 조건 조합

- 위에서 나열한 조건을 조합해서 다양한 Query를 작성할 수 있음
- 조합 순서 SELECT FROM WHERE ORDER BY LIMIT

예)

```
mysql> SELECT id, name FROM mytable

->  WHERE id < 4 AND name LIKE '%i%'

-> ORDER BY name DESC

-> LIMIT 2;
```

실습 - 데이터 검색

- 연습문제1: model_num 이 7700 으로 시작하는 로우(Row) 검색하기
- 연습문제2: name 이 i7 인 로우(Row) 검색하기
- 연습문제3: model_type 이 카비레이크 인 로우(Row) 를 1개만 검색하기(LIMIT 사용)

```
SELECT * FROM mytable WHERE model_num LIKE '7700%'
SELECT * FROM mytable WHERE name LIKE '%i7%'
SELECT * FROM mytable WHERE model_type LIKE '%카바레이크%' LIMIT 1
```

4.1.3 데이터 수정

- 테이블에 저장된 데이터를 수정하는 작업 - 기본 문법 (UPDATE) 1. 보통 WHERE 조건문과 함께 쓰여서, 특정한 조건에 맞는 데이터만 수정하는 경우가 많음

```
mysql> UPDATE 테이블명 SET 수정하고 싶은 컬럼명 = '수정하고 싶은 값' WHERE 특정 컬럼 = '값';
```

2. 다수의 컬럼 값을 수정할 수도 있음

```
mysql> UPDATE 테이블명 SET 수정하고 싶은 컬럼명1 = '수정하고 싶은 값',
수정하고 싶은 컬럼명2 = '수정하고 싶은 값', 수정하고 싶은 컬럼명3 = '수정하고 싶은 값' WHERE 특정 컬럼 < '값';
```

예)

```
mysql> SELECT * FROM mytable;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  3 | i7   | 7700      | Kaby Lake  |
+----+------+-----------+------------+
mysql> UPDATE mytable SET name = 'i5', model_num = '5500' WHERE id = 3;
mysql> SELECT * FROM mytable;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  3 | i5   | 5500      | Kaby Lake  |
+----+------+-----------+------------+
```

4.1.4 데이터 삭제

- 테이블에 저장된 데이터를 삭제하는 작업 - 기본 문법 (DELETE) 1. 보통 WHERE 조건문과 함께 쓰여서, 특정한 조건에 맞는 데이터만 삭제하는 경우가 많음

```
mysql> DELETE FROM 테이블명 WHERE 특정 컬럼 = '값';
```

2. 테이블에 저장된 모든 데이터를 삭제할 수도 있음

```
mysql> DELETE FROM 테이블명;
```

예)

```
mysql> SELECT * FROM mytable;
+----+------+-----------+------------+
| id | name | model_num | model_type |
+----+------+-----------+------------+
|  3 | i5   | 5500      | Kaby Lake  |
+----+------+-----------+------------+
1 row in set (0.00 sec)
mysql> DELETE FROM mytable WHERE id = 3;
Query OK, 1 row affected (0.01 sec)
mysql> SELECT * FROM mytable;
Empty set (0.00 sec)
```

> 실습 - 테이블 수정, 데이터 수정, 검색
>
> 사전조건: 이미 테이블에는 웹페이지(http://www.enuri.com/list.jsp?cate=070701) 에서 1위 ~ 5위까지 데이터가 입력되어 있어야 함
>
> 1. lowest_price(컬럼명) INT UNSIGNED(데이터타입) 으로 컬럼 추가
> 2. 최종 다음과 같은 모습으로 mytable 구조가 되어야 함

```
mysql> desc mytable;
+--------------+------------------+------+-----+---------+----------------+
| Field        | Type             | Null | Key | Default | Extra          |
+--------------+------------------+------+-----+---------+----------------+
| id           | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| name         | varchar(20)      | NO   |     | NULL    |                |
| model_num    | varchar(10)      | NO   |     | NULL    |                |
| model_type   | varchar(10)      | NO   |     | NULL    |                |
| lowest_price | int(10) unsigned | YES  |     | NULL    |                |
+--------------+------------------+------+-----+---------+----------------+
```

3. 웹페이지(http://www.enuri.com/list.jsp?cate=070701) 에서 1위 ~ 5위까지 lowest_price 값 수정하기

- lowest_price 값은 정품 최저 가격으로 입력하기

- 예: 222,180원 -> 222180

- 연습문제1: lowest_price 이 300000 이하인 로우(Row) 중에서 name과 model_num만 검색하기

. 이하는 같거나, 작은 값을 의미하고, 조건으로는 <= 와 같이 작성한다.

- 연습문제2: lowest_price 이 400000 이상인 로우(Row) 만 검색하기

. 이상은 같거나, 큰 값을 의미하고, 조건으로는 >= 와 같이 작성한다.

```
ALTER TABLE mytable ADD COLUMN lowest_price INT UNSIGNED;

UPDATE mytable SET lowest_price = 347100 WHERE id = 1;
...

SELECT name, model_num FROM mytable WHERE lowest_price < 300000;
```

5. SQL DCL(Data Control Language) 이해 및 실습

5-1 사용자 계정 만들기 / 권한 부여하기

root 계정 모든 호스트에서 mysql 접속 허용 설정하기

$ mysql -u root -p # mysql DB root 계정으로 접속

mysql> use mysql;
mysql> GRANT ALL PRIVILEGES ON . to 'root'@'%' IDENTIFIED BY 'korea123';
mysql> flush privileges;
mysql> exit

5-1-1 root 계정으로 접속, 사용자 계정 생성

```
$ sudo mysql -u root

mysql> use mysql;
mysql> show tables;

\- 계정 조회 하기
mysql> select host, user, authentication_string from user;
+-----------+------------------+-------------------------------------------+
| host      | user             | authentication_string                     |
+-----------+------------------+-------------------------------------------+
| localhost | root             |                                           |
| localhost | mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| localhost | mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| localhost | debian-sys-maint | *73778B62FC4E24667F7228F5B5D0095285A00764 |
+-----------+------------------+-------------------------------------------+

\- 사용자 계정(rapa01) 만들기
mysql> create user 'rapa01'@'%' identified by '1234';
mysql> select host, user, authentication_string from user;
+-----------+------------------+-------------------------------------------+
| host      | user             | authentication_string                     |
+-----------+------------------+-------------------------------------------+
| localhost | root             |                                           |
| localhost | mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| localhost | mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| localhost | debian-sys-maint | *73778B62FC4E24667F7228F5B5D0095285A00764 |
| %         | rapa01           | *3CD53EE62F8F7439157DF288B55772A2CA36E60C |
+-----------+------------------+-------------------------------------------+

[참고]
\- 로컬에서만 접속 가능한 userid 생성
mysql> create user 'userid'@localhost identified by '비밀번호';

\- 모든 호스트에서 접속 가능한 userid 생성
mysql> create user 'userid'@'%' identified by '비밀번호';

\- 사용자 비밀번호 변경
mysql> SET PASSWORD FOR 'userid'@'%' = '신규비밀번호';

\- 사용자 삭제
$ sudo mysql -u root -p
mysql> use mysql;
mysql> drop user 'userid'@'%';
```

5-1-2 생성한 사용자 계정으로 DB 접속

```
$ mysql -u rapa01 -p

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)

mysql> CREATE TABLE mytable (
    -> id INT UNSIGNED NOT NULL AUTO_INCREMENT,
    -> name VARCHAR(50) NOT NULL,
    -> modelnumber VARCHAR(15) NOT NULL,
    -> series VARCHAR(30) NOT NULL,
    -> PRIMARY KEY(id)
    -> );
ERROR 1046 (3D000): No database selected
```

- 아직은 할 수 있는게 없음.

5-1-3 사용자 계정에 권한 부여하기

db root 계정으로 접속하여 권한 부여하기

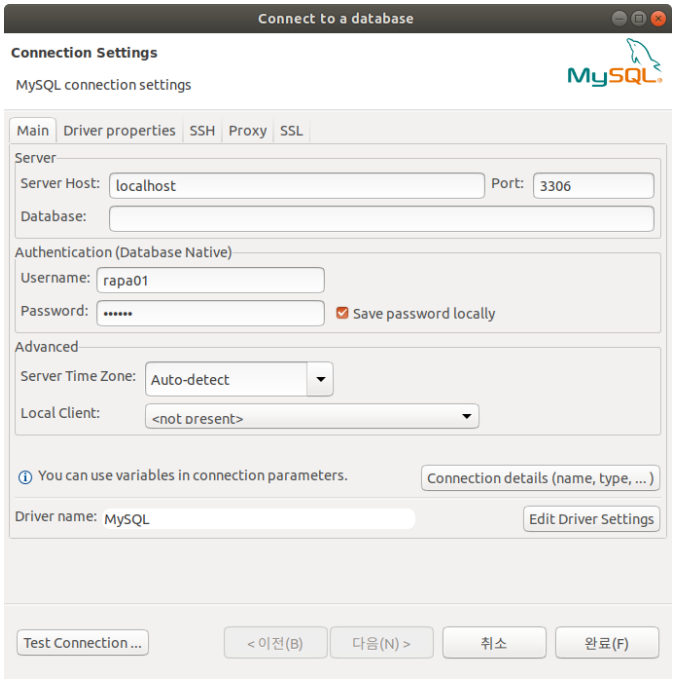
```
$ sudo mysql -u root -p

mysql> grant all privileges on \*.\* to 'rapa01'@'%';
```

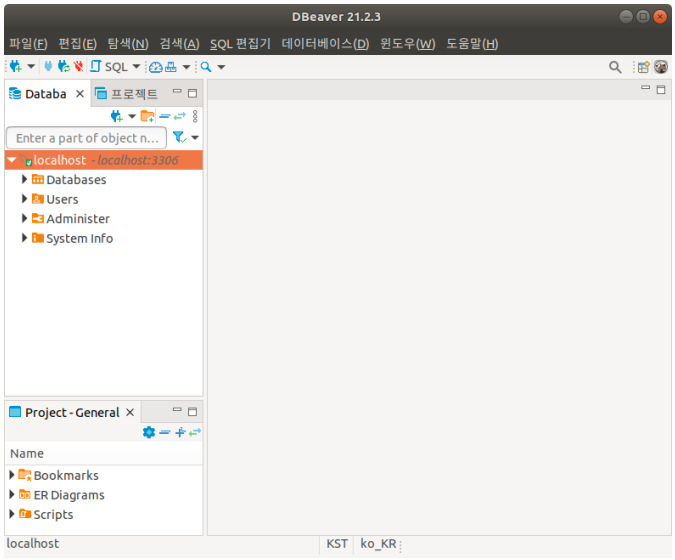
```
\- 사용자 계정 권한 조회
mysql> show grants for rapa01;
+-------------------------------------------+
| Grants for rapa01@%                       |
+-------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'rapa01'@'%'  |
+-------------------------------------------+
1 row in set (0.00 sec)

\- root 계정의 권한 조회
mysql> show grants;
+-------------------------------------------------------------------+
| Grants for root@localhost                                         |
+-------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION  |
| GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION        |
+-------------------------------------------------------------------+
```

5-1-4 사용자 계정으로 접속하여 권한 부여 확인하기

```
$ mysql -u rapa01 -p

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| bestproducts       |
| mydb               |
| mysql              |
| performance_schema |
| sqltest            |
| student_mgmt       |
| sys                |
+--------------------+
```

5-1-5 Dbeaver(DB 클라이언트)로 접속하기





실습 - 사용자 확인, 추가, 비밀번호 변경, 삭제

```
$ sudo mysql -u root -p
mysql> use mysql;
mysql> create user '사용자ID'@'%' identified by '비밀번호';
mysql> select host, user from user;
mysql> SET PASSWORD FOR '만들고싶은ID'@'%' = '신규비밀번호';
mysql> exit;
$ sudo mysql -u 사용자ID -p
mysql> exit;
$ sudo mysql -u root -p
mysql> use mysql;
mysql> drop user '사용자ID'@'%';
mysql> select host, user from user;
mysql> exit;
$ sudo mysql -u 사용자ID -p
에러가 나와야 함
```

5.2. mysql 접속 허용 설정[참고]

```
1) 현재 부여된 권한 확인하기
mysql> SHOW GRANTS for 아이디;
예) SHOW GRANTS for 'cozlab'@'%'

2) 로컬에서만 접속 허용
mysql> GRANT ALL ON DATABASE.TABLE to 'root'@localhost;

3) 특정 권한만 허용
mysql> GRANT SELECT, UPDATE ON DATABASE.TABLE to 'root'@localhost;


옵션 상세
(1) ALL – 모든 권한 / SELECT, UPDATE – 조회, 수정 권한등으로 권한 제한 가능
    예) GRANT INSERT,UPDATE,SELECT ON *.* TO 'username'@'localhost';
(2) DATABASE.TABLE : 특정 데이터베이스에 특정 테이블에만 권한을 줄 수 있음
    \*.\* : 모든 데이터베이스에 모든 테이블 권한을 가짐
(3) root – 계정명
(4) 1234 – 계정 비밀번호
```

6. pymysql 모듈 이해 및 실습

6.1 pymysql 라이브러리 소개 및 설치

- mysql을 python에서 사용할 수 있는 라이브러리
  (pymysql 라이브러리 이외에도 MySQLdb(Mysql-pytion), MySQL connector 등 다양한 라이브러리 존재)
- 이 중에서 설치가 가장 쉬운 라이브러리
- 설치
  - python -m pip install PyMySQL
  - pip install PyMySQL
- 일반적인 mysql 핸들링 코드 작성 순서
  1. PyMySql 모듈 import
  2. pymysql.connect() 메소드를 사용하여 MySQL에 연결
     - 호스트명, 포트, 로그인, 암호, 접속할 DB 등을 파라미터로 지정

3. MySQL 접속이 성공하면, Connection 객체로부터 cursor() 메서드를 호출하여 Cursor 객체를 가져옴
4. Cursor 객체의 execute() 메서드를 사용하여 SQL 문장을 DB 서버에 전송
5. SQL 쿼리의 경우 Cursor 객체의 fetchall(), fetchone(), fetchmany() 등의 메서드를 사용하여
   서버로부터 가져온 데이타를 코드에서 활용
6. 삽입, 갱신, 삭제 등의 DML(Data Manipulation Language) 문장을 실행하는 경우,
   INSERT/UPDATE/DELETE 후 Connection 객체의 commit() 메서드를 사용하여 데이타를 확정
7. Connection 객체의 close() 메서드를 사용하여 DB 연결을 닫음

- PyMySql 모듈 import

In [ ]:
```
import pymysql
```

- pymysql.connect() 메소드를 사용하여 MySQL에 연결
  - 호스트명, 포트, 로그인, 암호, 접속할 DB 등을 파라미터로 지정
  - 주요 파라미터
    - host : 접속할 mysql server 주소
    - port : 접속할 mysql server 의 포트 번호
    - user : mysql ID
    - passwd : mysql ID의 암호
    - db : 접속할 데이터베이스
    - charset='utf8' : mysql에서 select하여 데이타를 가져올 때 한글이 깨질 수 있으므로 연결 설정에 넣어줌

In [ ]:
```
conn = pymysql.connect(host='localhost', port=3306, user='rapa01',
                       passwd='1234', db='ecommerce', charset='utf8')
```

In [ ]:
```
conn
```

1. MySQL 접속이 성공하면, Connection 객체로부터 cursor() 메서드를 호출하여 Cursor 객체를 가져옴
2. Cursor 객체의 execute() 메서드를 사용하여 SQL 문장을 DB 서버에 전송

- 테이블 생성
  - Cursor Object 가져오기: cursor = db.cursor()
  - SQL 실행하기: cursor.execute(SQL)
  - 실행 mysql 서버에 확정 반영하기: conn.commit()

In [ ]:
```
cursor = conn.cursor()
```

In [ ]:
```
cursor
```

- cursor 는 control structure of database (연결된 객체로 봐도 됨)

In [ ]:
```
sql = """
    CREATE TABLE product (
        PRODUCT_CODE VARCHAR(20) NOT NULL,
        TITLE VARCHAR(200) NOT NULL,
        ORI_PRICE INT,
        DISCOUNT_PRICE INT,
        DISCOUNT_PERCENT INT,
        DELIVERY VARCHAR(2),
        PRIMARY KEY(PRODUCT_CODE)
    );
"""
```

- SQL 실행 (Cursor 객체의 execute() 메서드를 사용하여 INSERT, UPDATE 혹은 DELETE 문장을 DB 서버에 보냄)

In [ ]:
```
cursor.execute(sql)
```

- 삽입, 갱신, 삭제 등이 모두 끝났으면 Connection 객체의 commit() 메서드를 사용하여 데이타를 Commit

In [ ]:
```
conn.commit()
```

- Connection 객체의 close() 메서드를 사용하여 DB 연결을 닫음

In [ ]:
```
conn.close()
```

패턴으로 익히는 pymysql

In [ ]:
```
# 1. 라이브러리 가져오기
import pymysql

# 2. 접속하기
conn = pymysql.connect(host='localhost', port=3306, user='rapa01',
                       passwd='1234', db='ecommerce', charset='utf8')

# 3. 커서 가져오기
cursor = conn.cursor()

# 4. SQL 구문 만들기 (CRUD SQL 구문 등)
sql = '''
    CREATE TABLE product (
        PRODUCT_CODE VARCHAR(20) NOT NULL,
        TITLE VARCHAR(200) NOT NULL,
        ORI_PRICE INT,
        DISCOUNT_PRICE INT,
        DISCOUNT_PERCENT INT,
        DELIVERY VARCHAR(2),
        PRIMARY KEY(PRODUCT_CODE)
    );
'''

# 5. SQL 구문 실행하기
cursor.execute(sql)

# 6. DB에 Complete 하기
conn.commit()

# 7. DB 연결 닫기
conn.close()
```

- 데이터 삽입(INSERT)
  - Cursor Object 가져오기: cursor = conn.cursor()
  - SQL 실행하기: cursor.execute(SQL)
  - 실행 mysql 서버에 확정 반영하기: conn.commit()

In [ ]:
```
import pymysql

conn = pymysql.connect(host='127.0.0.1', port=3306, user='rapa01',
                       passwd='1234', db='ecommerce', charset='utf8')

cursor = conn.cursor()

for index in range(10):
    product_code = 215673140 + index + 1
    sql = """INSERT INTO product VALUES('""" + str(product_code) + """',
    '스위트바니 여름신상5900원~롱원피스티셔츠/긴팔/반팔', 23000, 6900, 70, 'F'); """
    print (sql)
    cursor.execute(sql)

conn.commit()
conn.close()
```

- 데이터 조회(SELECT)
  - Cursor Object 가져오기: cursor = conn.cursor()
  - SQL 실행하기: cursor.execute(SQL)
  - mysql 서버로부터 데이터 가져오기: fetch 메서드 사용
    - fetchall(): Fetch all the rows
    - fetchmany(size=None): Fetch several rows
    - fetchone(): Fetch the next row

In [ ]:
```
data = ((1, 2, 3, 4), (5, 6, 7, 8))
for item in data:
    print (item[1])
```

In [ ]:
```
import pymysql

conn = pymysql.connect(host='localhost', port=3306, user='rapa01',
                       passwd='1234', db='ecommerce', charset='utf8')
cursor = conn.cursor()
sql = "SELECT * FROM product"
cursor.execute(sql)
result = cursor.fetchone()
print(record)
conn.close()
```

- 데이터 수정(UPDATE)
  - Cursor Object 가져오기: cursor = conn.cursor()
  - SQL 실행하기: cursor.execute(SQL)
  - 실행 mysql 서버에 확정 반영하기: conn.commit()

달리샵린넨원피스 뷔스티에 썸머 가디건 코디전 33,000원 9,900원 70

```
UPDATE product SET
    TITLE='달리샵린넨원피스 뷔스티에 썸머 가디건 코디전',
    ORI_PRICE=33000,
    DISCOUNT_PRICE=9900,
    DISCOUNT_PERCENT=70
    WHERE PRODUCT_CODE='215673141'
```

In [ ]:

```
# 1. 라이브러리 가져오기
import pymysql

# 2. 접속하기
conn = pymysql.connect(host='localhost', port=3306, user='rapa01',
                       passwd='1234', db='ecommerce', charset='utf8')
# 3. 커서 가져오기
cursor = conn.cursor()

# 4. SQL 구문 만들기
SQL = """
UPDATE product SET
    TITLE='달리샵린넨원피스 뷔스티에 썸머 가디건 코디전',
    ORI_PRICE=33000,
    DISCOUNT_PRICE=9900,
    DISCOUNT_PERCENT=70
    WHERE PRODUCT_CODE='215673141'
"""

# 5. SQL 구문 실행하기
cursor.execute(SQL)

# 6. commit 하기
conn.commit()

# 7. close 하기
conn.close()
```

- 데이터 삭제(DELETE)
  - Cursor Object 가져오기: cursor = conn.cursor()
  - SQL 실행하기: cursor.execute(SQL)
  - 실행 mysql 서버에 확정 반영하기: conn.commit()

In [ ]:

```
DELETE FROM product WHERE PRODUCT_CODE='215673142'
```

In [ ]:

```
import pymysql

conn = pymysql.connect(host='localhost', port=3306, user='rapa01',
                       passwd='1234', db='ecommerce', charset='utf8')

cursor = conn.cursor()

SQL = """DELETE FROM product WHERE PRODUCT_CODE='215673142'"""

cursor.execute(SQL)

conn.commit()

conn.close()
```

In [ ]:

```

```