

# Structured Query Language for MySQL

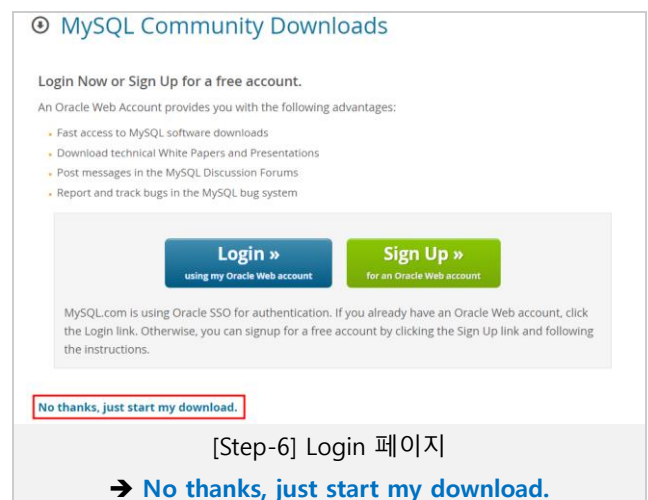
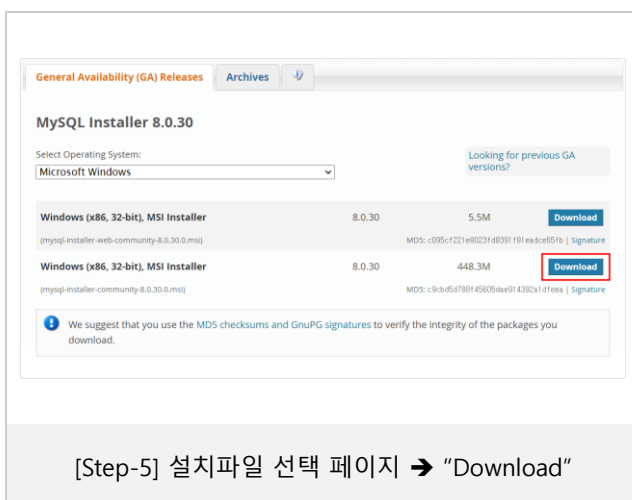
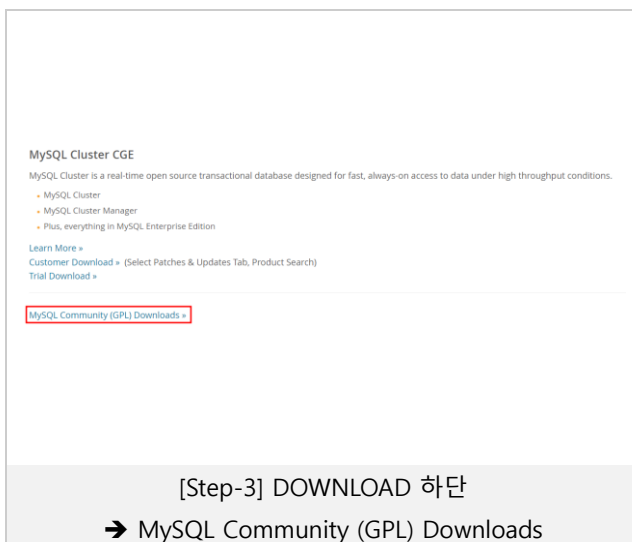
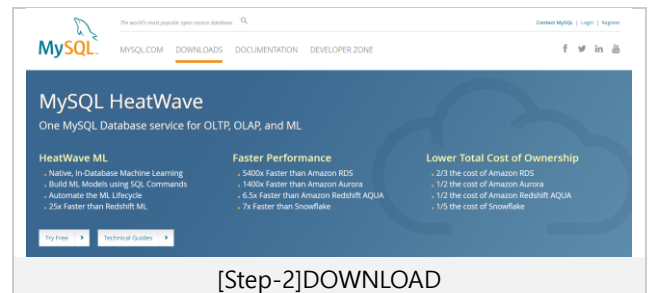
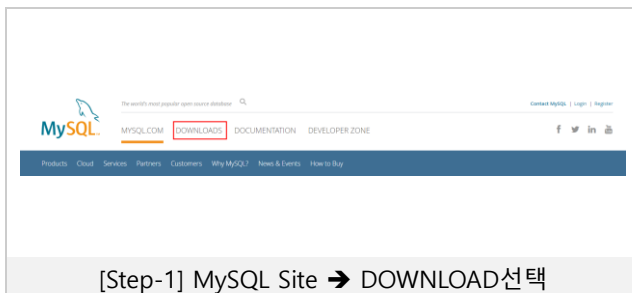
## I. Database 설치 및 개발환경 설정

### 1. Database 설치 및 환경설정

#### 1) MySQL Community Downloads : <https://www.mysql.com/>

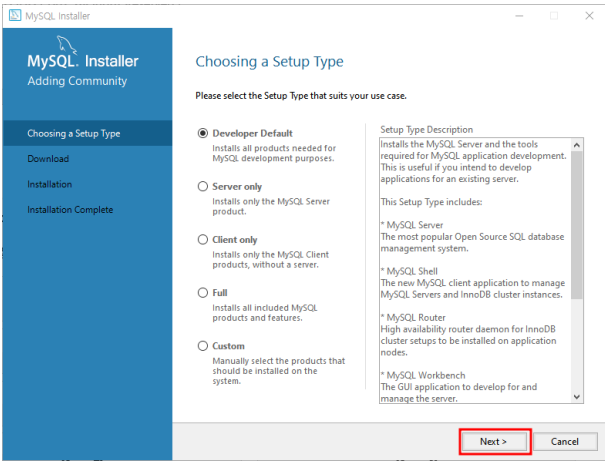
다운로드를 받기위한 아래 홈페이지의 단계적 순서 및 홈페이지의 구성과 다운로드 받을 "MySQL Community"의 버전 등은 사전 예고없이 변경될 수 있습니다.

"MySQL Community"의 2022년 9월 14일 현재 최신버전은 "8.0.30.0" 입니다.



## 2) MySQL Community의 설치 및 환경설정

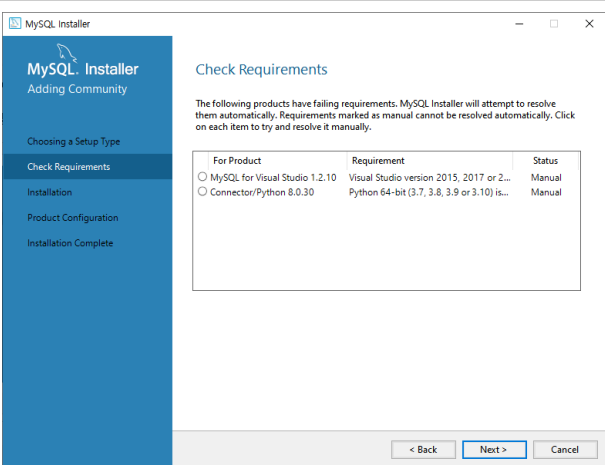
다운로드 파일(mysql-installer-community-8.0.30.0)은 버전에 따라 달라질 수 있습니다.



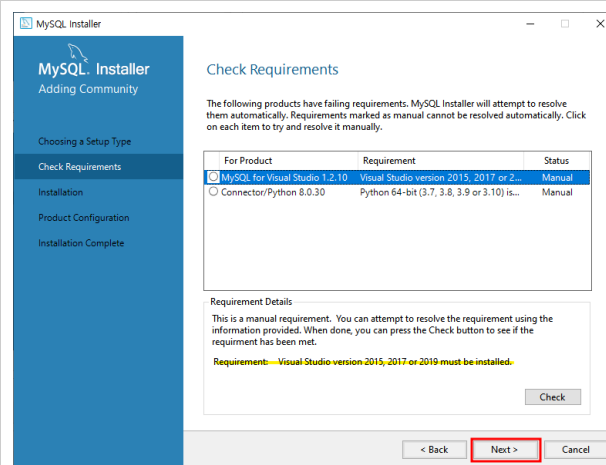
The screenshot shows the 'Choosing a Setup Type' window. The 'Developer Default' option is selected. The 'Next >' button is highlighted with a red box.

- **Developer Default ...** 개발에 필요한 서버 및 도구  
MySQL 서버, MySQL셸, MySQL 라우터, MySQL 워크벤치, Visual Studio용 MySQL, MySQL 커넥터, 예제, 문서
- **Server only ...** MySQL 서버만 설치
- **Client only ...** MySQL 서버와 예제, 문서 등 제외  
Developer Default에서 MySQL 서버, 예제, 문서를 제외한 개발에 필요한 사항만 설치
- **Full ...** MySQL과 관련된 모든 사항 설치  
Developer Default에 설명서 등을 추가하여 설치
- **Custom ...** 사용자 선택 설치  
사용자가 필요한 사항을 선택하여 설치

**[Step-1] Developer Default 선택 → "Next"**



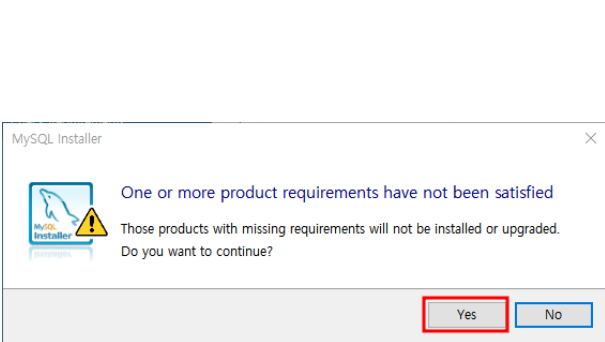
The screenshot shows the 'Check Requirements' window. A table lists requirements for Visual Studio and Python. The 'Next >' button is highlighted with a red box.



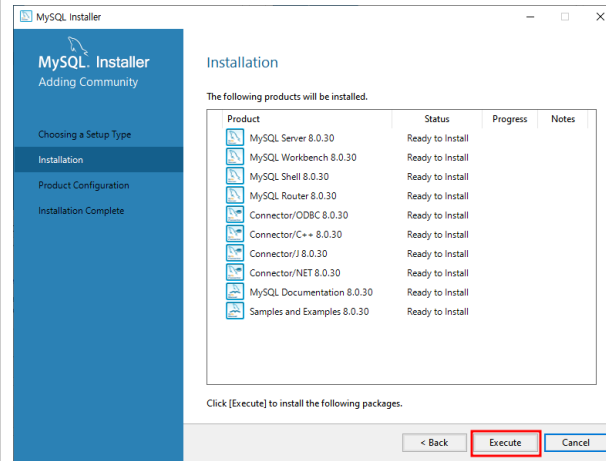
The screenshot shows the 'Check Requirements' window with a detailed view of the requirement for Visual Studio. The 'Next >' button is highlighted with a red box.

**[Step-2] 설치되지 않는 도구 확인**

**[Step-3] 설치 못하는 도구 상세 내용 → "Next"**



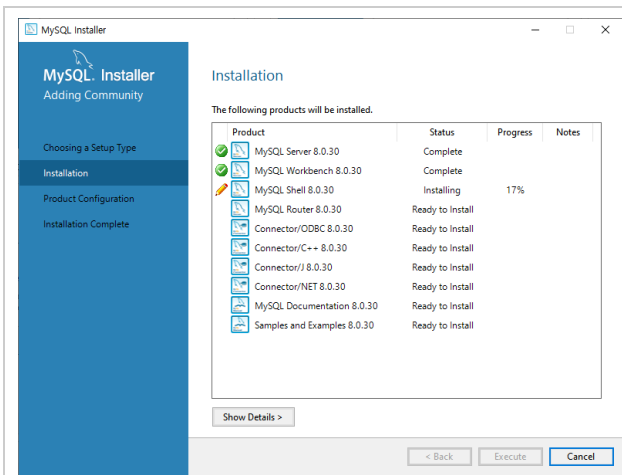
The screenshot shows a warning dialog box stating that some requirements are not satisfied. The 'Yes' button is highlighted with a red box.



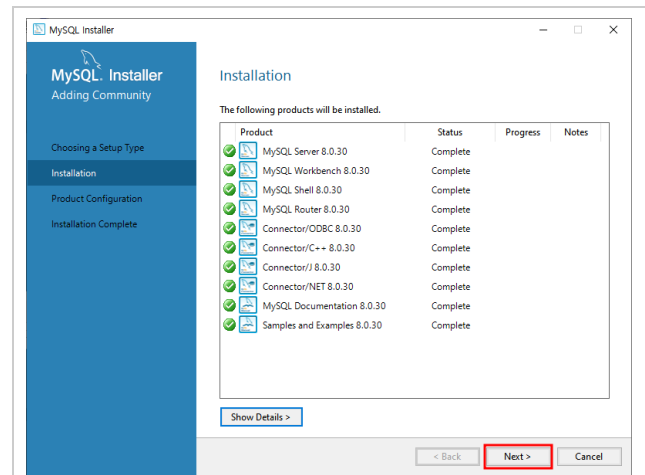
The screenshot shows the 'Installation' window. A table lists products to be installed, all with a status of 'Ready to install'. The 'Execute' button is highlighted with a red box.

**[Step-4] "step-3"의 도구 미 설치 후 설치 사용자 확인**

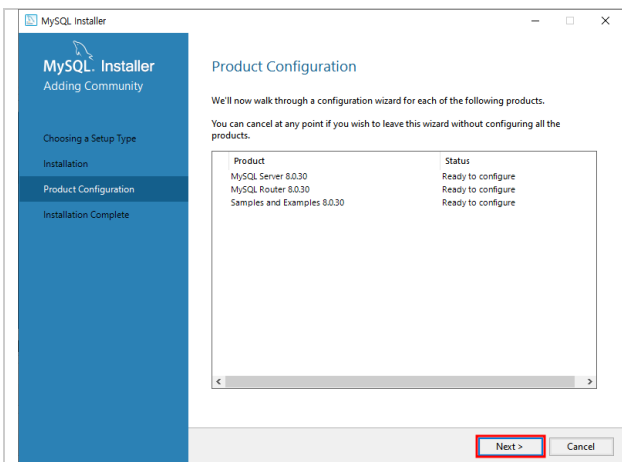
**[Step-5] 설치되는 제품 → "Execute"**



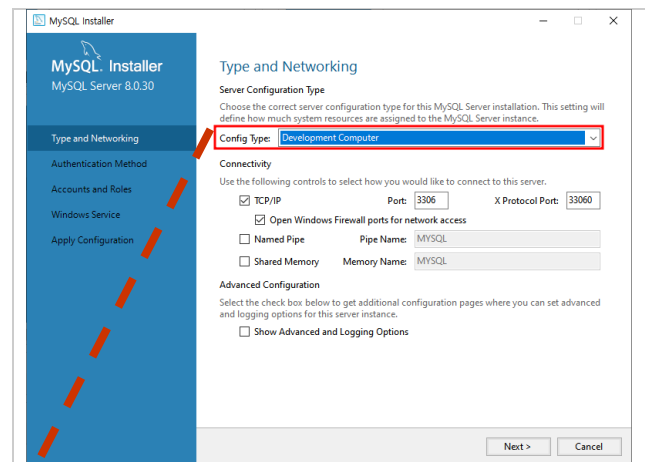
[Step-6] 포함된 제품 설치 중



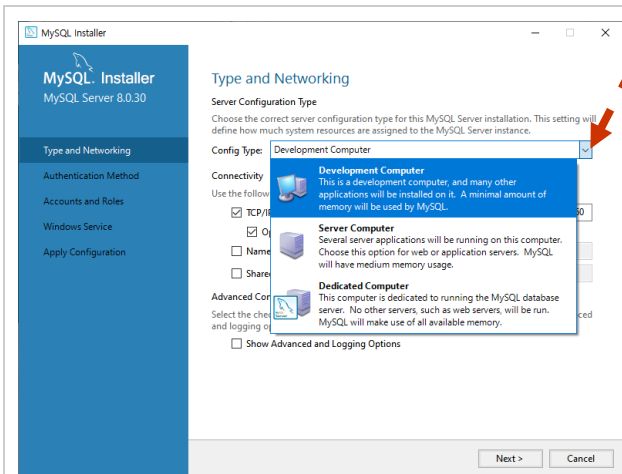
[Step-7] 포함된 제품 설치완료 → "Next"



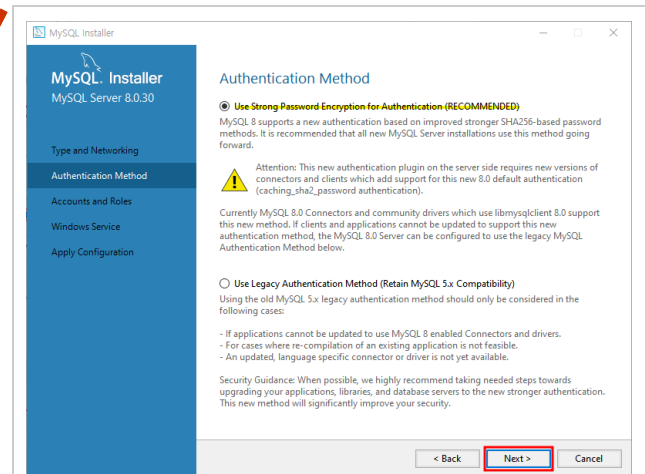
[Step-8] 설치된 제품 중 설정이 필요한 제품 → "Next"



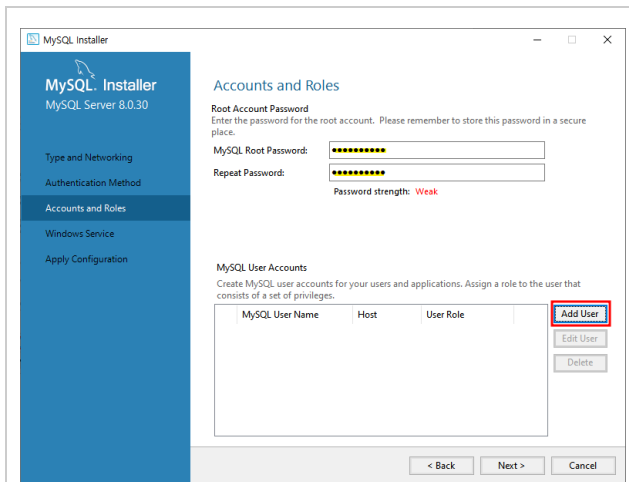
[Step-9] 사용 유형 및 Networking



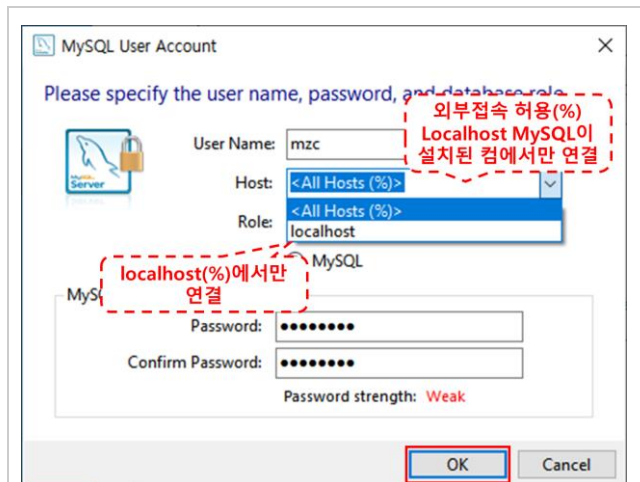
[Step-10] Config Type → Development Computer  
→ "Next"



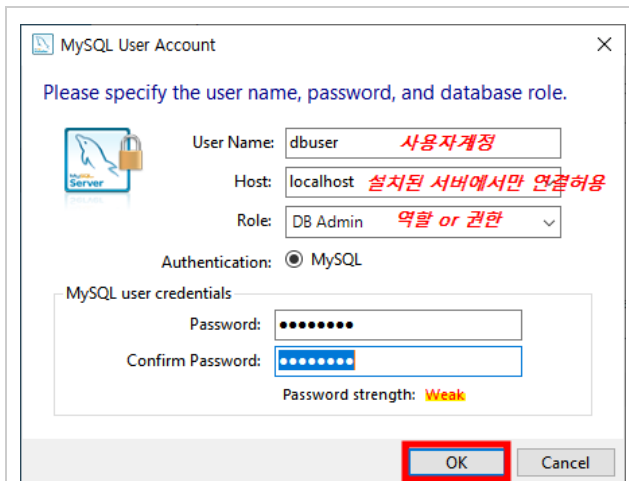
[Step-11] 비밀번호 유형 → 강력한 보안 → "Next"



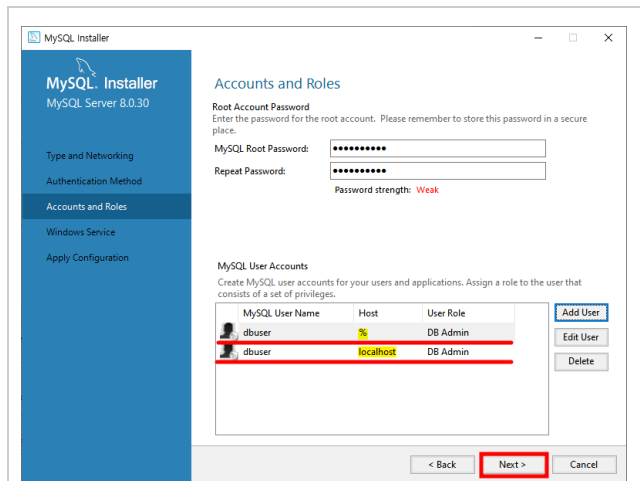
[Step-12] root 비밀번호 설정 → "Add User"



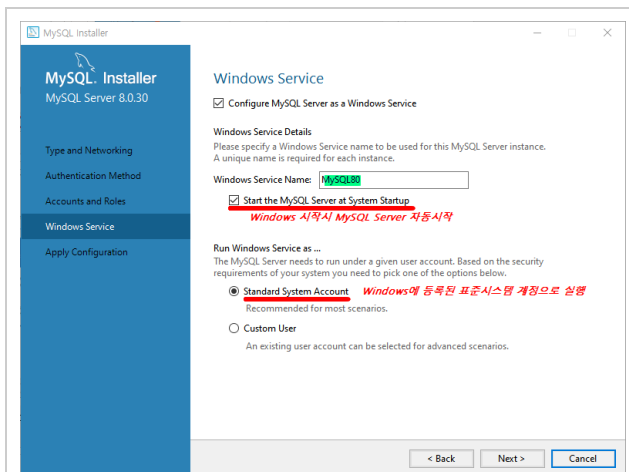
[Step-13] 사용자 등록 - 외부접속허용



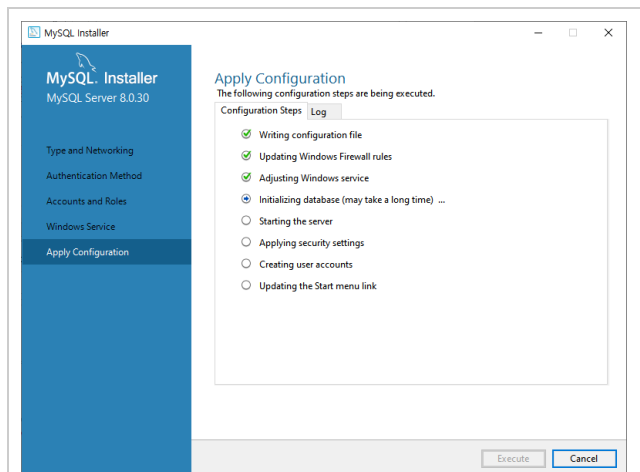
[Step-14] 사용자 등록 - 설치된 서버에서만 연결허용



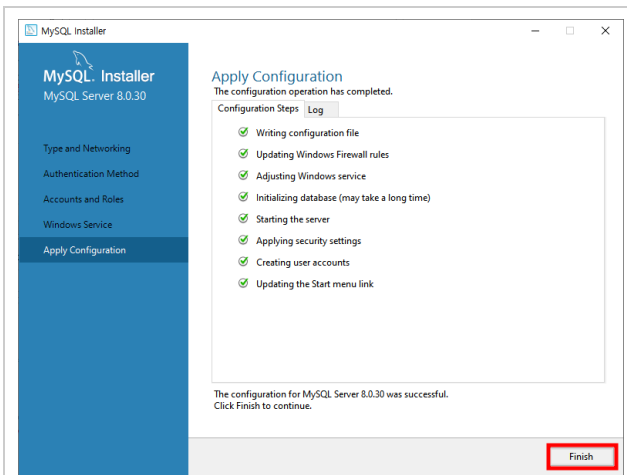
[Step-15] 등록된 사용자 계정 정보



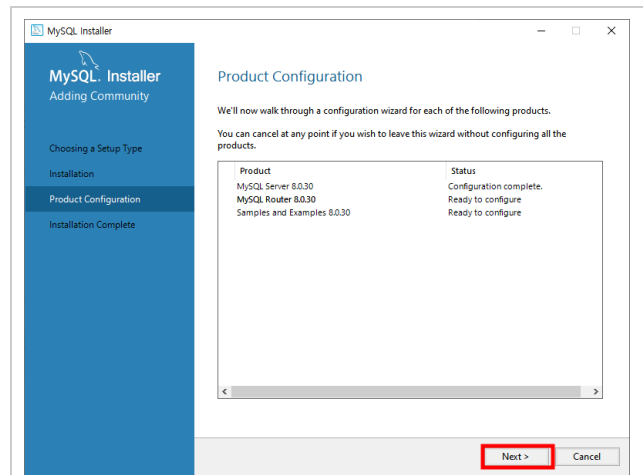
[Step-16] MySQL Server 자동실행 및 실행계정 선택



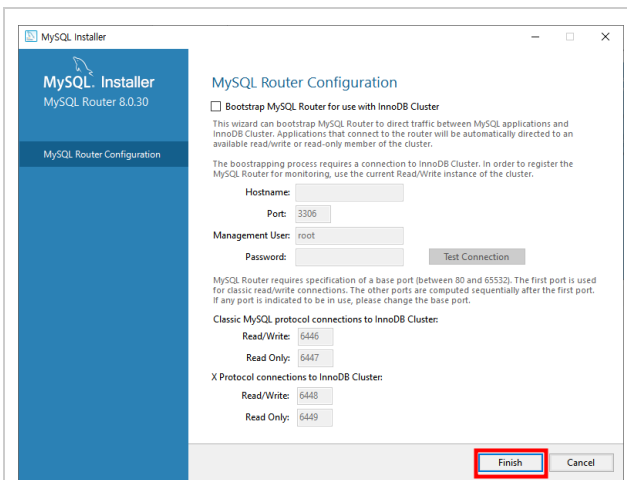
[Step-17] 환경설정 실행 중



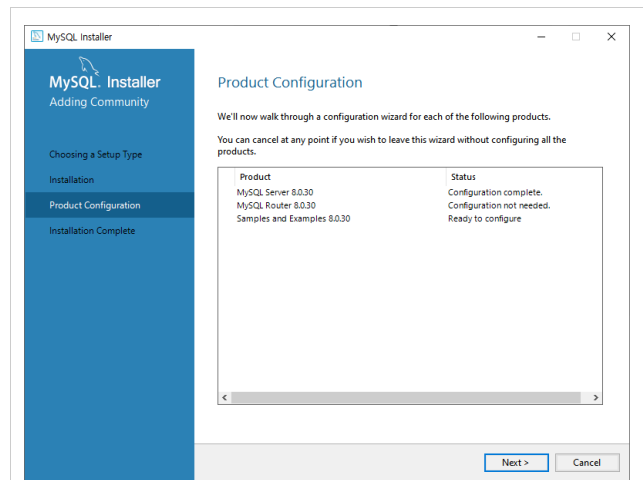
[Step-18] 환경설정 실행완료



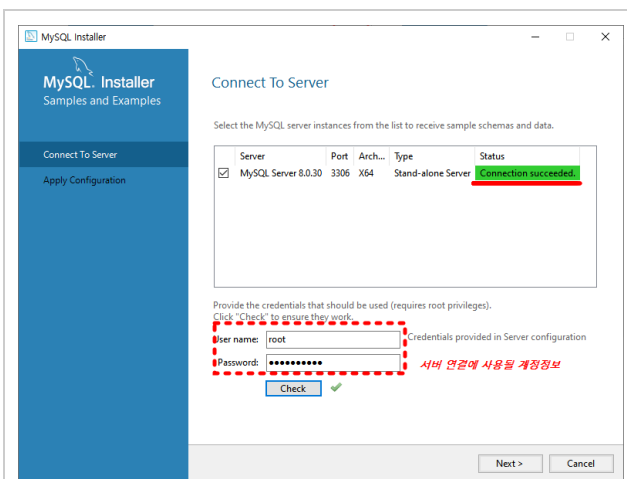
[Step-19] 제품의 환경설정 중



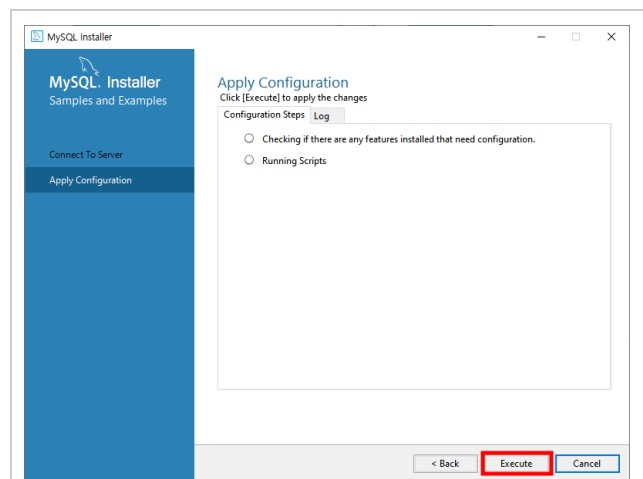
[Step-20] 제품 MySQL Router 환경설정



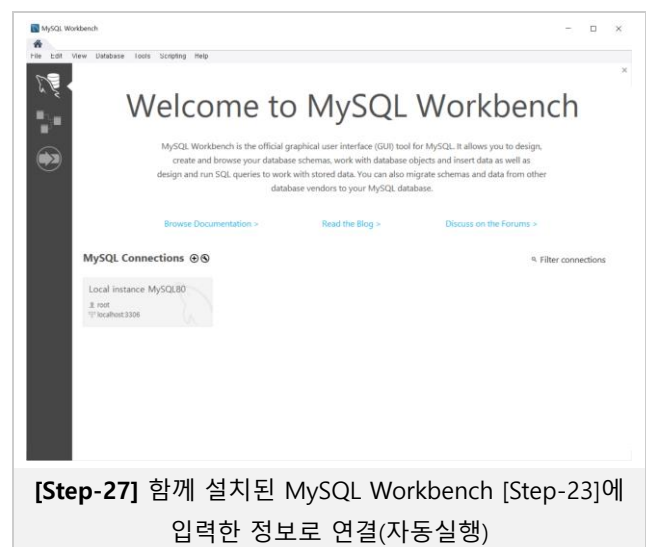
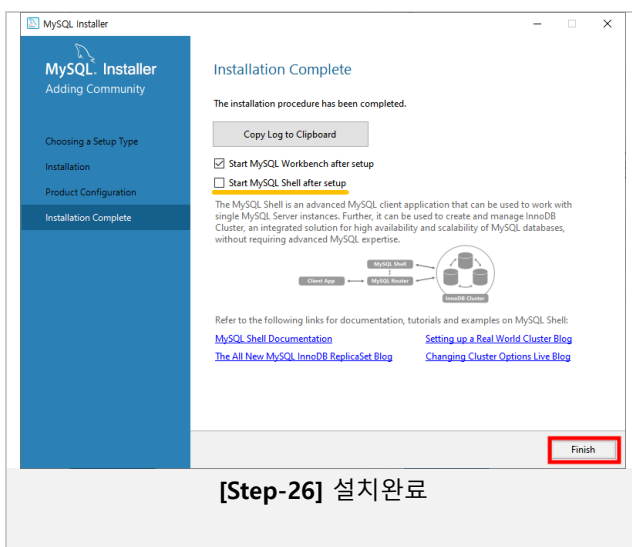
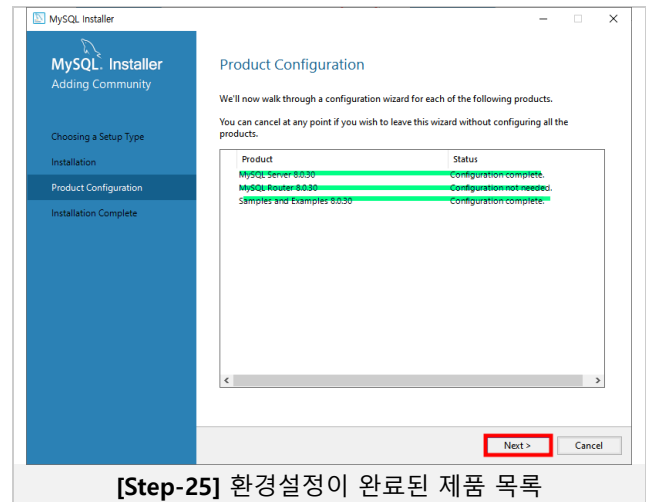
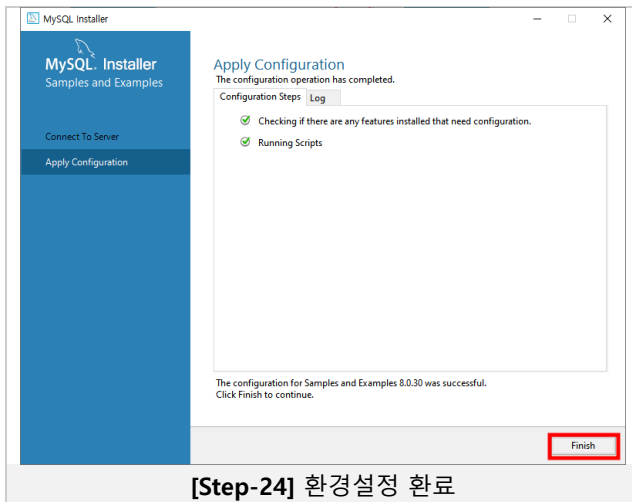
[Step-21] 제품의 환경설정



[Step-22] 서버연결을 위한 정보



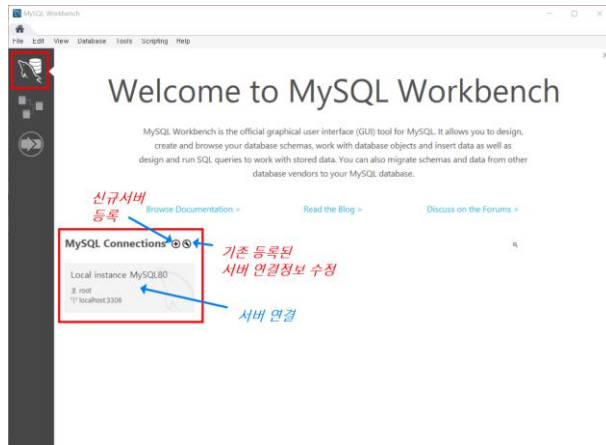
[Step-23] 환경설정 실행



### 3) MySQL Workbench 환경설정

MySQL Server에 연결하여 Database 프로그래밍을 하기 위해서는 **MySQL Workbench**를 통해 연결을 해야 합니다. 이를 위한 연결 설정은 다음과 같은 순서로 진행하시면 됩니다.

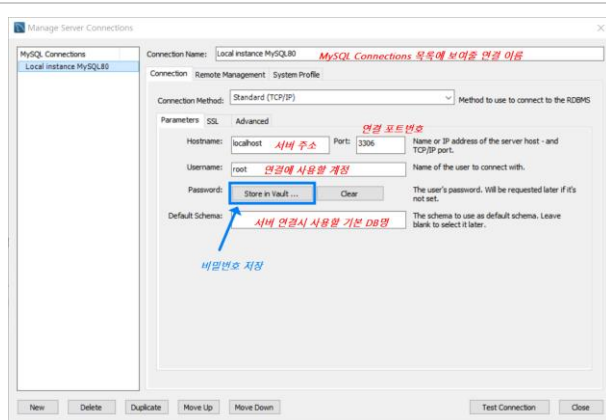
다만, MySQL Community의 설치할 때 **MySQL Workbench** 도 함께 설치가 되었다면 설치단계에서 이 연결 정보를 등록하고 설치 종료 후 입력된 정보를 이용하여 연결을 시도합니다.



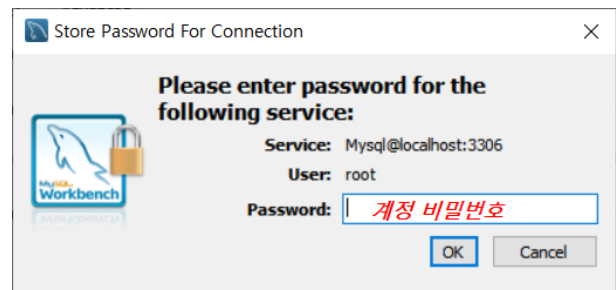
[Step-1] MySQL Workbench 실행화면



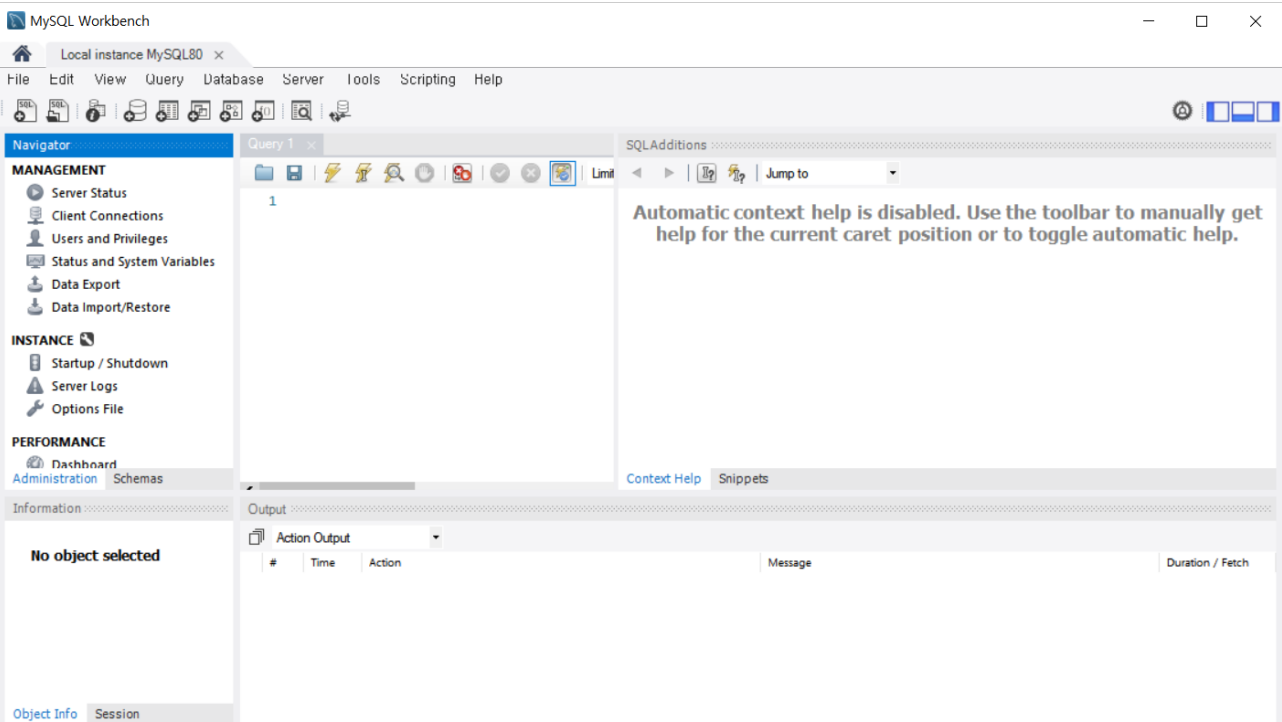
[Step-2] 신규 서버 등록 및 수정 실행



[Step-3] 연결할 MySQL Server 등록 및 수정



[Step-4] Store Password For Connection



[Step-5] MySQL Server 연결 화면



## ※ MySQL 주석문

1. '#'으로 시작하는 1개 라인
2. '--'(마이너스 기호 2번)으로 시작하는 1개 라인
3. 여러 개의 줄을 한번에 주석 처리 하고자 하는 경우엔 '/\*' ~ '\*/'와 같이 사용함

## II. 데이터 정의 언어 (DDL : Data Definition Language)

### 1. 데이터베이스 생성 및 삭제

CREATE DATABASE문은 MySQL Server에 Database를 생성해 줍니다.

#### 1) 형식

##### ① Database 생성

```
CREATE DATABASE <databasename>;
```

##### ② DATABASE 이름변경 : *RENAME DATABASE구문이 버전 5.1.23 이후부터 위험성으로 인해 제거됨*

###### [ 방법 1 ]

1. 새 DATABASE 생성
2. 기존 DATABASE의 내용(TABLE, PROCEDURE 등)들을 새 DATABASE로 옮김  
→ **RENAME TABLE `기존DB명`.TABLE명` TO `새DB명.새로운TABLE명` ;**
3. 옮길 PROCEDURE가 있다면 아래 쿼리를 통해 옮겨준음.  
→ **UPDATE mysql.proc SET DB = `기존DB명` WHERE DB = 새DB명` ;**
4. 정상적으로 새 DATABASE에 옮겨졌다면 기존 DATABASE를 삭제함(*반드시 삭제해야 하는 것은 아님*)  
→ **DROP DATABASE `기존DB명` ;**

###### [ 방법 1 ]

```
SELECT concat('RENAME TABLE ',TABLE_SCHEMA,',',TABLE_NAME,' TO ', 'test2.',TABLE_NAME,';')
FROM information_schema.tables
WHERE TABLE_SCHEMA LIKE 'test';
```

##### ③ Database 삭제

```
DROP DATABASE <databasename>;
```

#### 2) 실습예제

```
CREATE DATABASE study ;
DROP DATABASE style ;
```

## 2. 테이블 생성

### 1) 형식

#### ① 생성

##### [ 형식 ]

```
CREATE TABLE <TABLENAME> (
    <FIELD_NAME> integerType [ AUTO_INCREMENT ],
    <FIELD_NAME> dataType(len) [ NOT NULL ],
    <FIELD_NAME> dataType(len) [ DEFAULT <값> ],
    <FIELD_NAME> dataType(len) DEFAULT SYSDATE ,
    :
    PRIMARY KEY ( <FIELD_NAME> ) ,
    [ CONSTRAINT <제약조거이름> ] FOREIGN KEY ( <FIELD_NAME> )
    REFERENCES <TABLE_NAME> ( <FIELD_NAME> )
) ENGINE=InnoDB DEFAULT CHARSET=utf8 comment '테이블 설명' ;
```

##### [ 실습예제 ]

```
CREATE TABLE TEX1 (
    FIDX INT NOT NULL COMMENT '사업명 코드',
    FNAME NVARCHAR(20) NOT NULL COMMENT '사업명',
    FEMAIL VARCHAR(20) NOT NULL COMMENT '관리/감독 기관명',
    FPHONE VARCHAR(13) NOT NULL COMMENT '정부부처명',
    FADD CHAR(1) DEFAULT '1' COMMENT '사용여부',
    PRIMARY KEY (FPROJECT_CODE)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='사업정보' ;
```

#### ② 수정

##### [ 형식 ]

###### [ 새로운 필드 추가 ]

```
ALTER TABLE <TABLENAME> ADD <FIELD_NAME> dataType
```

###### [ 기존 필드의 삭제 ]

```
ALTER TABLE <TABLENAME> DROP <FIELD_NAME> ;
```

###### [ 필드 타입 변경 ]

```
ALTER TABLE <TABLENAME> MODIFY COLUMN <FIELD_NAME> dataType
```

##### [ 실습예제 ]

Column	Type	Default Value	Nullable	Character Set	Collation
◇ FADDR	varchar(50)	1	YES	utf8mb3	utf8mb3_genera...
◇ FEMAIL	varchar(20)		NO	utf8mb3	utf8mb3_genera...
◇ FIDX	int		NO		
◇ FNAME	varchar(20)		NO	utf8mb3	utf8mb3_genera...
◇ FPHONE	varchar(13)		NO	utf8mb3	utf8mb3_genera...

#### [ 새로운 필드 추가 ]

조건 : 위

#### [ 기존 필드의 삭제 ]

#### [ 필드 타입 변경 ]

### ③ 삭제

```
DROP TABLE <TABLENAME> ;
```

### 2) 예제

### 4) 데이터 유형

#### ① 문자형 데이터

데이터 유형	정의
<b>CHAR(n)</b>	고정 길이 데이터 타입(최대 255byte)- 지정된 길이보다 짧은 데이터 입력될 시 나머지 공간 공백으로 채워진다.
<b>VARCHAR(n)</b>	가변 길이 데이터 타입(최대 65535byte)- 지정된 길이보다 짧은 데이터 입력될 시 나머지 공간은 채워지지 않는다.
TINYTEXT(n)	문자열 데이터 타입(최대 255byte)
TEXT(n)	문자열 데이터 타입(최대 65535byte)

MEDIUMTEXT(n)	문자열 데이터 타입(최대 16777215byte)
LONGTEXT(n)	문자열 데이터 타입(최대 4294967295byte)
enum(string, string, ... )	
JSON	JSON 문자열 데이터 타입 - JSON 형태의 포맷을 꼭 준수해야 한다.

## ② 숫자형 데이터

데이터 유형	정의
TINYINT(n)	정수형 데이터 타입(1byte) -128 ~ +127 또는 0 ~ 255수 표현할 수 있다.
SMALLINT(n)	정수형 데이터 타입(2byte) -32768 ~ 32767 또는 0 ~ 65536수 표현할 수 있다.
MEDIUMINT(n)	정수형 데이터 타입(3byte) -8388608 ~ +8388607 또는 0 ~ 16777215수 표현할 수 있다.
INT(n)	정수형 데이터 타입(4byte) -2147483648 ~ +2147483647 또는 0 ~ 4294967295수 표현할 수 있다.
BIGINT(n)	정수형 데이터 타입(8byte) - 무제한 수 표현할 수 있다.
FLOAT(길이, 소수)	부동 소수형 데이터 타입(4byte) -고정 소수점을 사용 형태이다.
DECIMAL(길이, 소수)	고정 소수형 데이터 타입(고정(길이+1byte) -소수점을 사용 형태이다.
DOUBLE(길이, 소수)	부동 소수형 데이터 타입(8byte) -DOUBLE을 문자열로 저장한다.

## ③ 날짜형 데이터

데이터 유형	정의
DATE	날짜(년도, 월, 일) 형태의 기간 표현 데이터 타입(3byte)
TIME	시간(시, 분, 초) 형태의 기간 표현 데이터 타입(3byte)
DATETIME	날짜와 시간 형태의 기간 표현 데이터 타입(8byte)

TIMESTAMP	날짜와 시간 형태의 기간 표현 데이터 타입(4byte) -시스템 변경 시 자동으로 그 날짜와 시간이 저장된다.
YEAR	년도 표현 데이터 타입(1byte)

#### ④ 이진 데이터

데이터 유형	정의
BINARY(n) & BYTE(n)	CHAR의 형태의 이진 데이터 타입 (최대 255byte)
VARBINARY(n)	VARCHAR의 형태의 이진 데이터 타입 (최대 65535byte)
TINYBLOB(n)	이진 데이터 타입 (최대 255byte)
BLOB(n)	이진 데이터 타입 (최대 65535byte)
MEDIUMBLOB(n)	이진 데이터 타입 (최대 16777215byte)
LOB(n)	이진 데이터 타입 (최대 4294967295byte)

#### ⑤ 데이터형 선택방법

- TYPE은 작을수록 좋다
- TYPE은 단순한 게 좋다
- 가능 하면 NULL을 쓰지 말자

#### ⑥ 테이블 명세서

▶ TABLE NAME : TSCHOOL 학교교 기본 정보						
COLUMN	TYPE	NULL	KEY	DEFAULT	OTHER	COMMENT
FSCHOOL_IDX	INT		P/K			학교 일련번호 지정
FSCHOOL_NAME	NVARCHAR(40)					학교이름
FSCHOOL_ADDRES S	NVARCHAR(20)					학교 캠퍼스 주소
FSCHOOL_GRADE	TINYINT					학위구분

▶ 학위구분 : 고등검정(1), 고등(2), 학점전문학사(3), 전문(4), 학점학사(5), 정규대학(6), 대학원(7), 기타(9-해외, 고졸이하)
▶ 학교 캠퍼스 주소 : 경기도 00 시, 서울특별시 00 구
▶

## 4. 사용자 정의 함수

### 1) 환경설정 변경

MySQL의 기본적으로 함수의 생성을 불허하고 있다. 아래와 같이 현재 상태를 확인해 봅니다.

```
SHOW GLOBAL VARIABLES LIKE 'log_bin_trust_function_creators';
```

	Variable_name	Value
▶	log_bin_trust_function_creators	OFF

결과 같이 위와 같이 나올 경우 다음과 같이 "log\_bin\_trust\_function\_creators" 값을 "ON"으로 변경해주면 됩니다.

```
SET GLOBAL log_bin_trust_function_creators = 'ON';
```

### 2) 함수 만들기

```
DELIMITER $$
CREATE FUNCTION [함수 이름]([입력값 이름][데이터 타입], ...) RETURNS [결과값 데이터 타입]
BEGIN
    DECLARE [임시값 이름][데이터 타입];
    SET [임시값 이름] = [입력값 이름];
    쿼리문;
    RETURN 결과값
END $$
DELIMITER ; -- 세미콜론 앞에 공백을 주어야 함
```

### 3) 함수 삭제

```
DROP FUNCTION IF EXISTS <functionName>;
```

### 4) 기본 예제

```
DELIMITER $$
CREATE FUNCTION funEX1() RETURNS NVARCHAR(10)
BEGIN
    DECLARE msg NVARCHAR(10) DEFAULT null;
    SET msg = '함수생성 성공';
    RETURN msg;
END $$
DELIMITER ; -- 세미콜론 앞에 공백을 주어야 함
```

## DELIMITER \$\$

```
CREATE FUNCTION funEX2(kor TINYINT, eng TINYINT, mat TINYINT) RETURNS NVARCHAR(10)
BEGIN
    DECLARE tot SMALLINT          DEFAULT 0;
    DECLARE ave TINYINT           DEFAULT 0;
    DECLARE msg NVARCHAR(10)      DEFAULT 0;
    SET tot = kor + eng + mat;
    SET ave = tot / 3;
    IF ave >= 80 THEN
        SET msg = '우수 학생';
    ELSEIF ave >= 60 THEN
        SET msg = '승인 학생';
    ELSE
        SET msg = '재시험 대상자';
    END IF;
    RETURN msg;
END $$
```

**DELIMITER ;** -- 세미콜론 앞에 공백을 주어야 함

## 5) 실습 예제

1. 아래와 같이 테이블을 생성 합니다.

```
CREATE TABLE TEX1 (
    FIDX    INT          auto_increment COMMENT '일련번호',
    FNAME   NVARCHAR(20) NOT NULL      COMMENT '이름',
    FPHONE  VARCHAR(13)  NOT NULL      COMMENT '핸드폰',
    FKOR    TINYINT      NOT NULL      COMMENT '국어 성적',
    FENG    TINYINT      NOT NULL      COMMENT '영어 성적',
    FMAT    TINYINT      NOT NULL      COMMENT '수어 성적',
    PRIMARY KEY (FIDX)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='성적 테이블' ;
```

2. 함수를 만들어 데이터를 추가합니다.

3. 데이터 삽입이 성공되면 "성공", 삽입하지 못할 경우 '실패'를 출력 합니다.



## 5. 스토어 프로시저

프로시저는 일련의 쿼리를 하나의 함수처럼 실행하기 위한 쿼리의 집합이며, 매개변수를 받을 수 있고 반복적으로 사용할 수 있는 블록입니다.

### - 프로시저의 장점

- 하나의 요청으로 여러 SQL문을 실행할 수 있기 때문에 네트워크에 대한 부하를 줄일 수 있습니다.
- 안전합니다. DBA는 기본 데이터베이스 테이블에 접근할 수 있는 권한을 아무에게 부여하지 않고 응용 프로그램이 저장 프로시저에 접근할 수 있는 권한을 부여하는 방법을 사용합니다.

### - 프로시저의 단점

- 많은 프로시저들을 사용한다면 모든 연결의 메모리 사용량이 증가하기 때문에 CPU사용량이 증가하게 됩니다.
- 문자, 숫자열에 연산에 대한 프로시저를 사용하면 오히려 C, JAVA보다 느린 성능을 볼 수 있습니다.
- 디버깅 및 개발, 유지보수가 쉽지 않습니다.

## 1) 프로시저의 사용방법

### ① 프로시저의 생성

```
DELIMITER $$
CREATE PROCEDURE '프로시저명' (
  IN <parameter> <data type>,
  IN <parameter> <data type>,
  OUT <parameter> <data type>
)
BEGIN
  DECLARE <variable identifier> <data type> DEFAULT <data>;
  수행할 쿼리
  :
END $$
DELIMITER ;
```

- DELIMITER \$\$
  - 프로시저 구문과는 관련이 없는 명령어로, 표준 구분 기호인 세미콜론(;)을 다른 기로 (\$\$)로 변경합니다.
  - MySQL도구가 매번 각 문장을 실행하는 것보다 서버에 프로시저를 통과시키는 것이 중요하기 때문입니다.
- END \$\$
  - 프로시저의 끝을 표시할 때 DELIMITER에서 정의한 구분기호를 사용합니다.
- DELIMITER ;
  - 구분기호를 다시 세미콜론으로 변경하기 위해 사용합니다.
- IN
  - 파라미터를 받을 경우에는 IN을 사용합니다.
- OUT
  - 결과값을 내보내는 경우에는 OUT을 사용합니다.
- *<parameter>*
  - 파라미터 이름
- DECLARE
  - 변수를 선언하는 경우 사용합니다.
- *<variable identifier>*
  - 변수 이름

## 2) 실습예제

### ① 테이블 생성



## 6. 내장함수

MySQL은 내장함수를 제공합니다. 제어함수, 문자열 함수, 수학 함수, 날짜 및 시간 함수 등 비교적 많은 종류의 함수가 있습니다.

### 1) 제어함수

Function	Description
case 문	