

모두의 MySQL

2024.01

모두의 MySQL

▣ 강의 및 실습은 Windows 10 64bit 기반으로 진행됩니다.

▣ 아래 항목 중 3개이상 보유 시 본 과정의 목표를 효과적으로 달성할 수 있습니다.

- EXCEL 중급자 이상
- 데이터, 숫자에 대한 재미, 흥미, 관심, 더 알고자 하는 호기심
- 뉴스의 데이터 시각화, 데이터 저널리즘에 대한 관심 및 보고서 작성 경험
- 기업 IT 부서의 데이터베이스에 대한 관심 및 경험
- 인공지능, 빅데이터, 데이터분석에 대한 관심 및 경험

✓ 학습목표

- SQL 데이터베이스에서 기본적인 문법을 사용하여 데이터를 추출할 수 있습니다.
- SQL 데이터베이스에 있는 정보를 활용하여 자유자재로 검색, 추출, 수정할 수 있습니다.

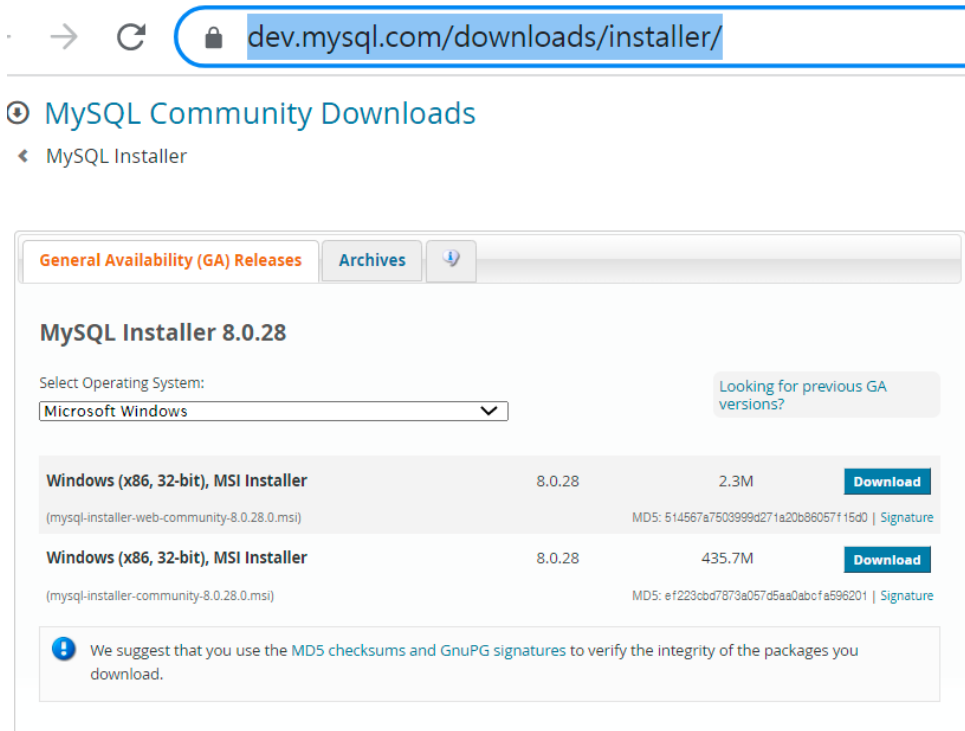
✓ 학습대상

- 기업의 DB에 무엇이 들어있는지 궁금하셨던 분
- IT 부서에 요청하느니 '차라리 내가 다운받아서 추출해서 보고 싶다'라고 생각하셨던 분
- 데이터, 자동화, 효율성, 인공지능, 4차산업혁명에 관심이 있으신 분

I. MySQL 환경설정

데이터베이스 프로그램 설치 및 환경설정

1. Mysql Installer 다운로드



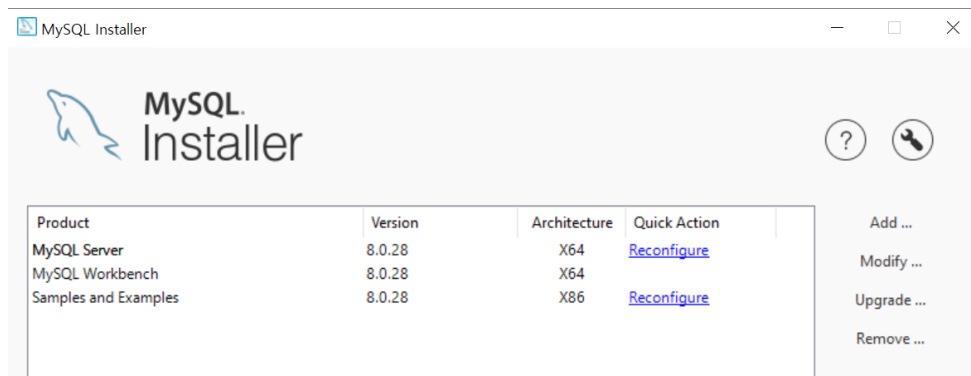
MySQL 프로그램 다운로드 주소:

<https://dev.mysql.com/downloads/installer/>

또는 구글에서 'MySQL Installer' 입력

[화면 좌측 하단의 No thanks, just start my download 선택]

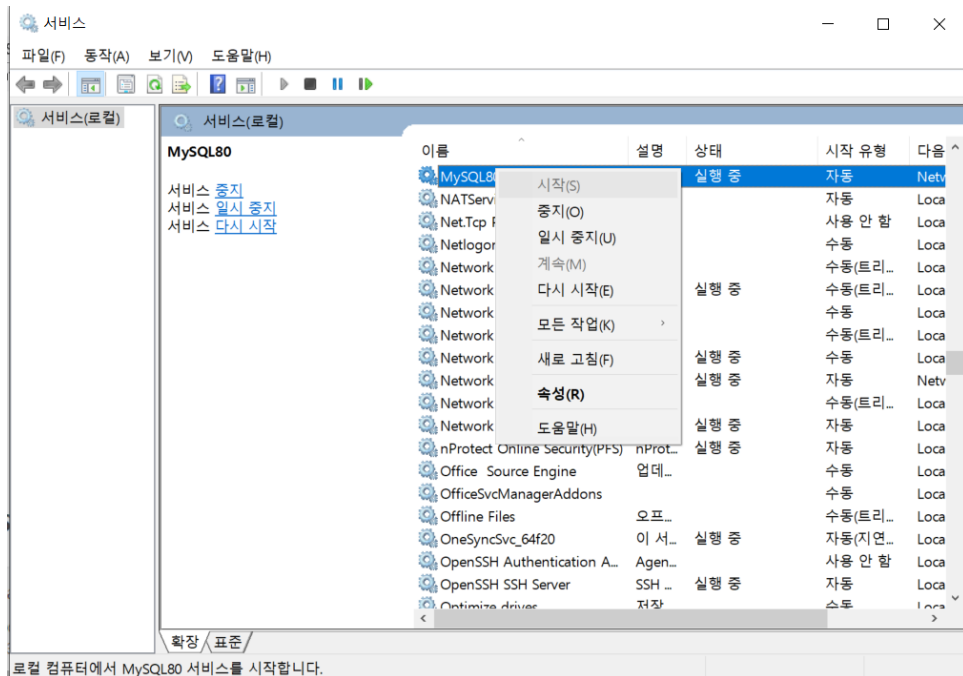
MySQL Workbench 설치



- MySQL Server ☒
- MySQL WorkBench ☒
- Samples and Examples ☒

를 선택합니다.

MySQL 서버 On/Off 방법



이번에는 MySQL 서버 데몬을 켜고/끄는 (ON/OFF) 방법을 알아보시다.

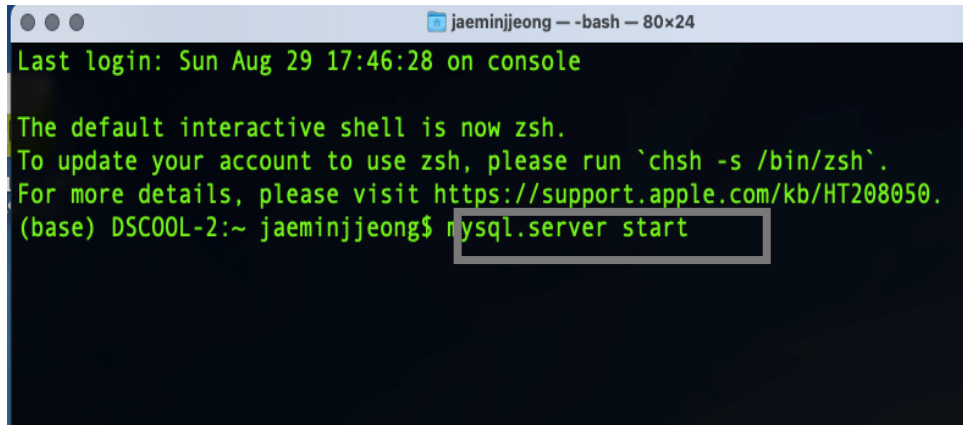
Windows:

시작메뉴 검색창에 'SERVICE' 입력 (services.msc)

-> MYSQL80 (마우스 오른쪽 클릭 -> '시작')

BACKGROUND RUN - MySQL 서버 데몬

Mac 서버 구동 방법



```
jaeminjeong ~ -bash - 80x24
Last login: Sun Aug 29 17:46:28 on console

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) DSC00L-2:~ jaeminjeong$ mysql.server start
```

Command Prompt (bash)에

\$ mysql.server start 입력!!

Mac의 경우 - MySQL SERVER 8.0 +

MySQL WorkBench 다운로드 / 설치

\$ mysql.server start (시작)

\$ mysql.server stop (중단)

\$ mysql.server restart (재시작)

* 에러 메시지 - google.com 검색 또는 stackoverflow.com 검색!!

에러 메시지 해결 방법 - google 또는 stackoverflow.com 에

‘에러 메시지’를 검색 후 해당 내용을 찾아 보완

데이터셋 다운로드

DATA
공공데이터포털
GO . KR

[데이터찾기](#)
[국가데이터맵](#)
[데이터요청](#)
[데이터활용](#)
[정보공유](#)
[이용안내](#)

[목록등록관리시스템](#)
[로그인](#)
[회원가입](#)
 [사이트맵](#)
[ENGLISH](#)

데이터 상세

[f](#)
[t](#)
[y](#)
[URL 복사](#)

부산지방공공단체포일 선수정보

2021년도 경찰선수 성적정보안내(년도,선수번호,선수명,등급,출주회차수,출주일수,1위횟수,승률1위,2위횟수,연대율,3위횟수,3연대율,4위횟수,5위 횟수,6위횟수,7위횟수,8위횟수,9위횟수,상격,낙차포기,고장포기)

0

0

관심

파일데이터

오픈API

공공데이터활용지원센터는 공공데이터포털에 개방되는 3단계 이상의 오픈 포맷 파일데이터를 오픈 API(RestAPI 기반의 JSON/XML)로 자동변환하여 제공합니다. 오픈 API를 활용하기 위해서는 공공데이터포털 회원 가입 및 활용신청이 필요하며, 활용 관련 문의는 공공데이터활용지원센터로 연락주시기 바랍니다. 파일데이터는 로그인 없이 다운로드를 통해 이용하실 수 있습니다.

CSV

부산지방공공단체포일 선수정보

[다운로드](#)
[오픈신고 및 담당자 문의](#)

파일데이터 정보

메타데이터 다운로드

| | | | |
|---------|----------------------------|-----------|--------------|
| 파일데이터명 | 부산지방공공단체포일 선수정보, 202111231 | | |
| 분류체계 | 문화체육관광 - 체육 | 제공기관 | 부산지방공공단체포일 |
| 관리부서명 | 감사평가단 | 관리부서 전화번호 | 051-550-1641 |
| 보유근거 | 경찰경정법 | 수집방법 | |
| 업데이트 주기 | 연간 | 자기 등록 예정일 | 2023-02-28 |
| 메타유형 | 텍스트 | 전체 행 | 536 |

www.data.go.kr

첫 번째 올라온 데이터셋을
다운로드하겠습니다!!

검색창에 키워드 검색

데이터 → 다운로드

MySQL 환경설정 에서 다룬 내용

- MySQL SERVER, MySQL Workbench, Samples and Examples 설치!!
- Mysql server 실행하기!!

(Windows: 'SERVICES.msc' -> mysql80실행 / Mac: \$ mysql.server start 실행)

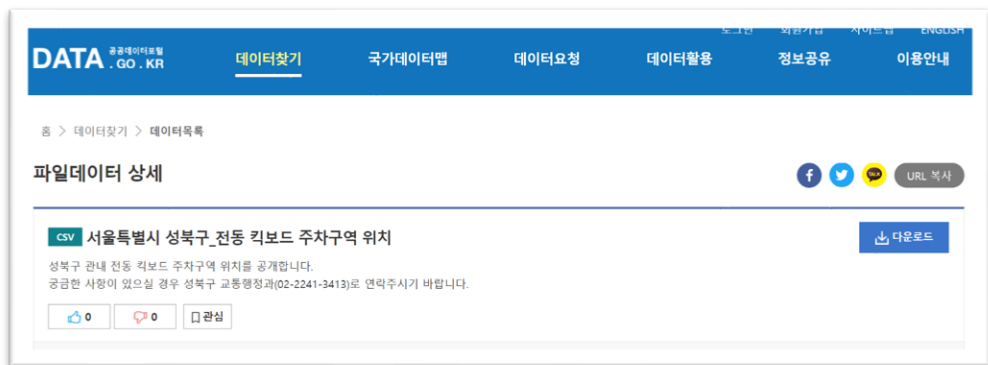
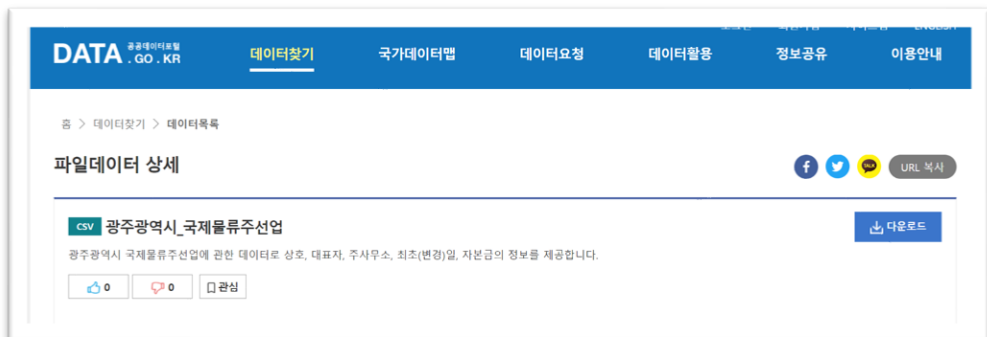
- Schema / 데이터베이스 만들기
- Table / 데이터 한 페이지 만들기 (Excel의 Sheet과 같은 개념!!)
- .csv 데이터 업로드하기!! (Data Import Wizard)

II. MySQL 기초 - INSERT, DELETE, CREATE TABLE문

이번 단원에서 다룰 내용

- MySQL 테이블 생성 (DB에서 표 만들기)
- 공공데이터포털에서 사용할 데이터 DOWNLOAD
- 다운로드한 공공데이터 MySQL에서 IMPORT
- 마우스 클릭으로 데이터 한 행 추가하기
- INSERT문으로 데이터 한 행 추가하기
- 마우스 클릭으로 데이터 한 줄 삭제하기
- DELETE문으로 데이터 한 줄 삭제하기
- CREATE TABLE 문으로 테이블 생성하기
- 두 번째 DATASET으로 같은 내용 실습하기!!

data.go.kr에서 데이터 다운로드



강의게시판 .csv파일 다운로드!

* cp949 오류시: a. 메모장으로 열어서 'ANSI'로 다시 저장

b. 엑셀에서 다른 이름으로 저장 ->

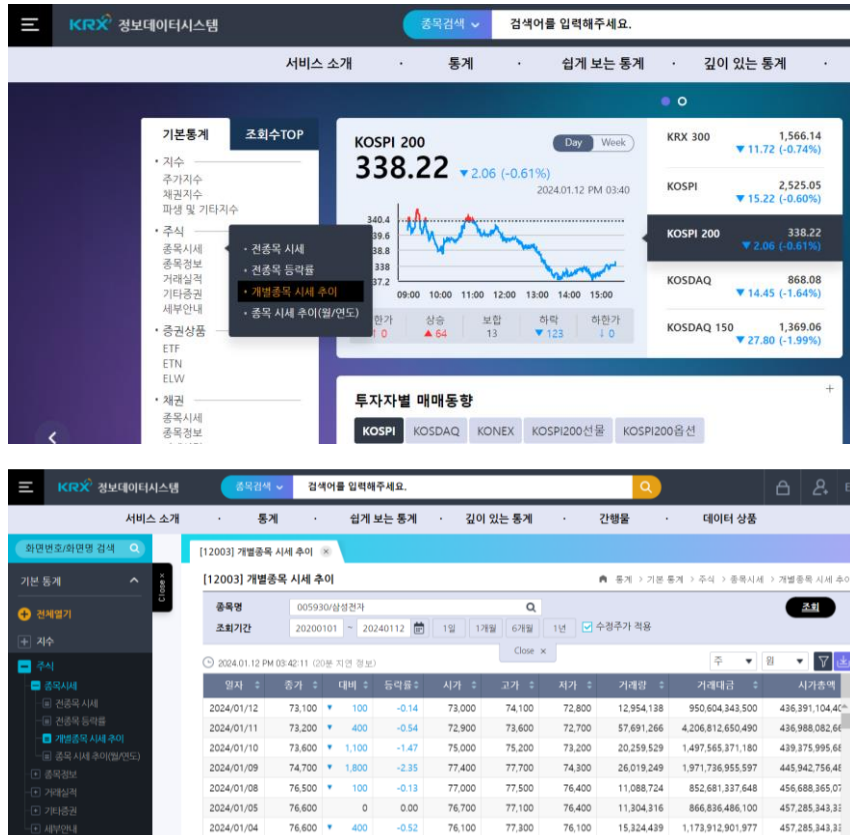
.csv (utf-8) [X]

.csv (Macintosh) [0] 선택 저장

삼성전자 주식 데이터 다운로드

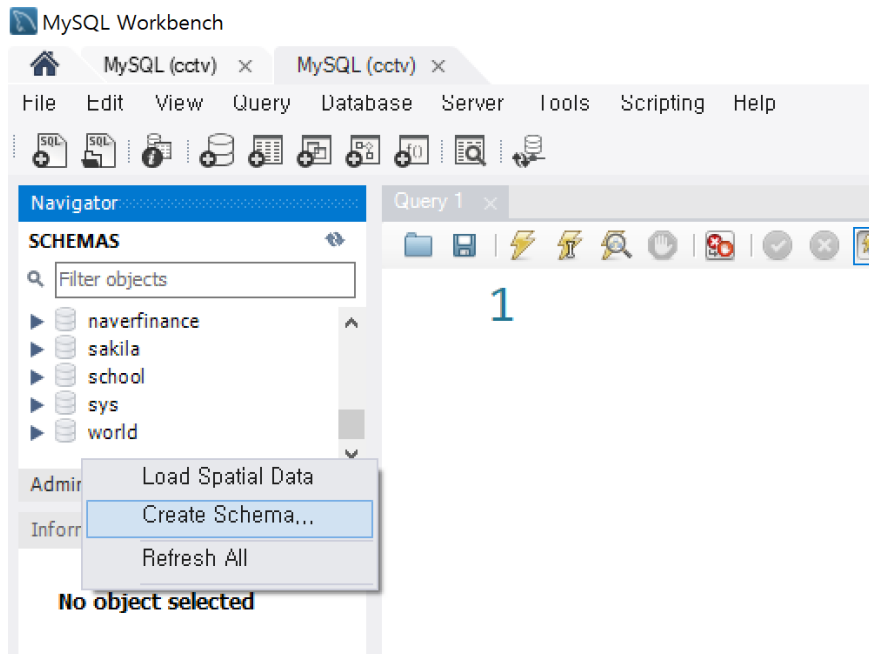
<http://data.krx.co.kr>

종목시세 -> 개별종목 시세 추이

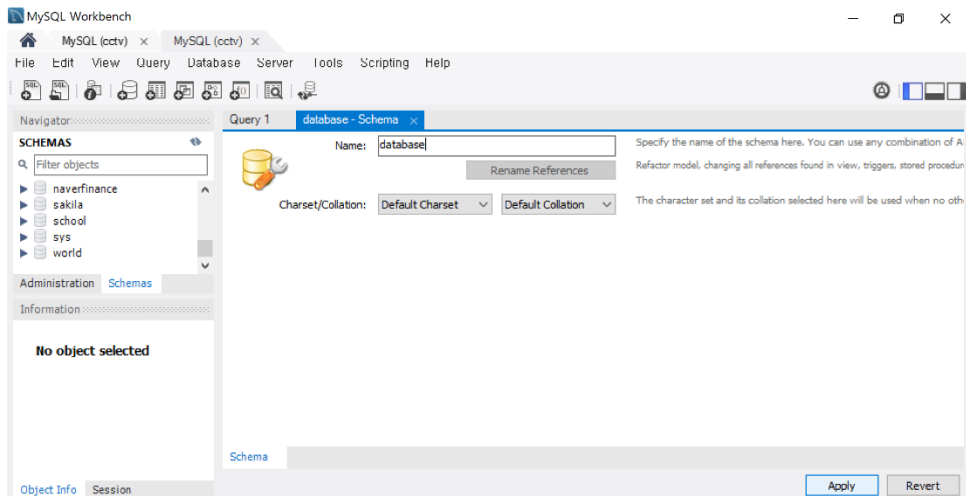


날짜 범위 설정 후, 다운로드 버튼 클릭하여 .csv로 다운로드합니다.

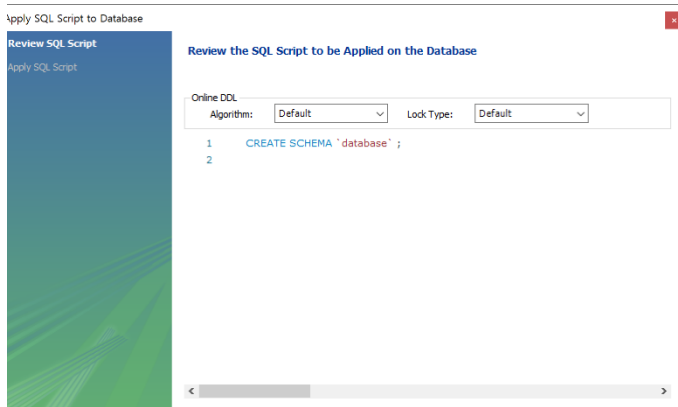
데이터베이스 생성하기 : CREATE DATABASE



좌측 Navigator -> 하단 Schema 탭 -> 우클릭 ->
'Create Schema'를 선택하여 DB를 생성합니다.



스키마(DB, 데이터베이스) 이름을 'database'로 입력합니다. Apply 버튼 클릭!!

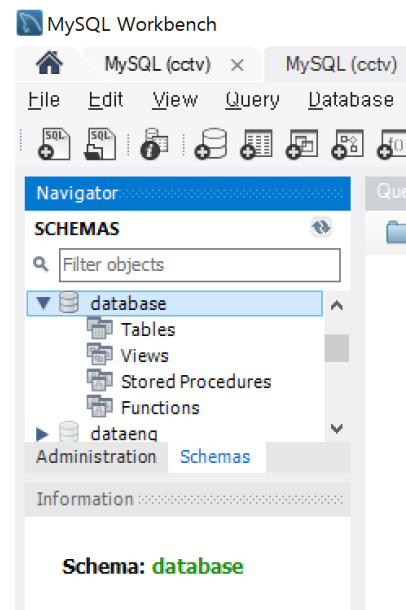
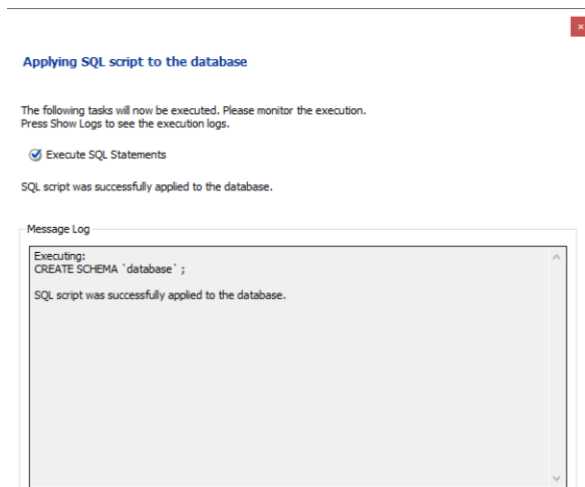


이 화면에 나오는 SQL문을 잘 살펴봅시다.

CREATE SCHEMA `database` ;

명령어를 MySQL Workbench가 자동으로 생성해서 실행,

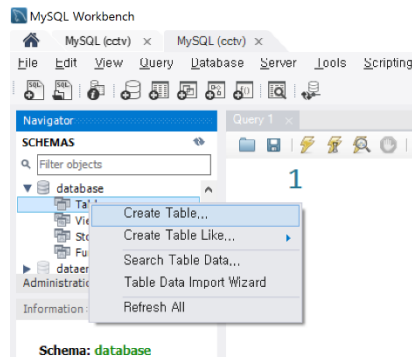
데이터베이스를 생성함을 알 수 있습니다.



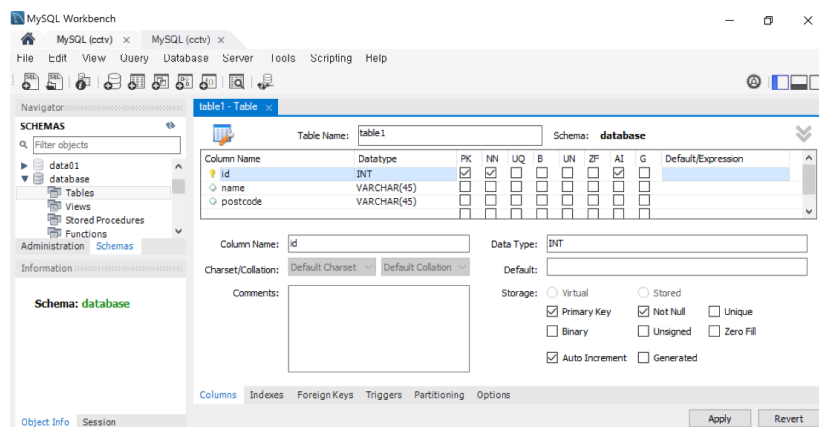
이제 'SCHEMAS' 우측의 Reload 버튼을 눌러서 새로고침 해 봅시다.

생성한 'database' 스키마(Database)가 보입니다.

테이블 생성



이번에는 DB 하위에 테이블 (TABLE)을 생성해 보겠습니다.
Database -> Table 우클릭 -> Create Table 선택



컬럼(column)을 일일이 입력해 줍니다.

| Column Name | Datatype | PK | NN | ... | AI |
|-------------|-------------|----|----|-----|----|
| Id | INT | v | v | ... | v |
| Name | VARCHAR(45) | | | ... | |
| Postcode | VARCHAR(45) | | | ... | |

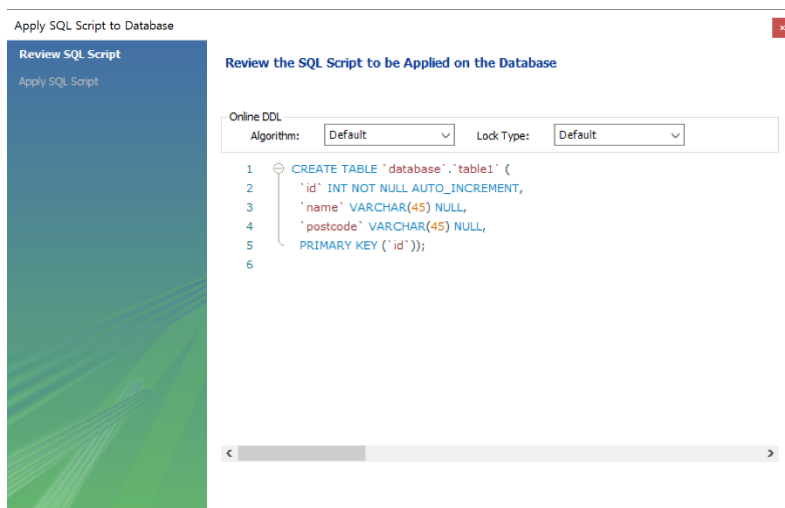
컬럼 3 개를 추가해줍니다. Id, name, postcode 세 개의 컬럼을

추가하고,

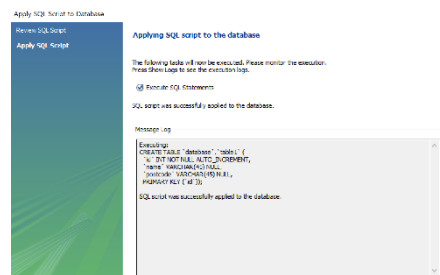
Datatype 은 각각 int, varchar(45), varchar(45)로 지정해 줍니다. int 는 정수, varchar(45)는 최대 45 자의 문자열까지 저장할 수 있다는 뜻입니다.

Id 컬럼의 옵션에는 PK, NN, AI 에 체크해줍니다.

- PK = Primary Key : index 와 같은 역할을 하는 고유번호. 서로 겹치지 않습니다.
- NN = Not Null : 빈칸으로 둘 수 없습니다.
- AI = Auto Increment : 사용자가 값을 입력하지 않아도 자동으로 +1 씩 더해 컴퓨터가 자동으로 입력해 줍니다.

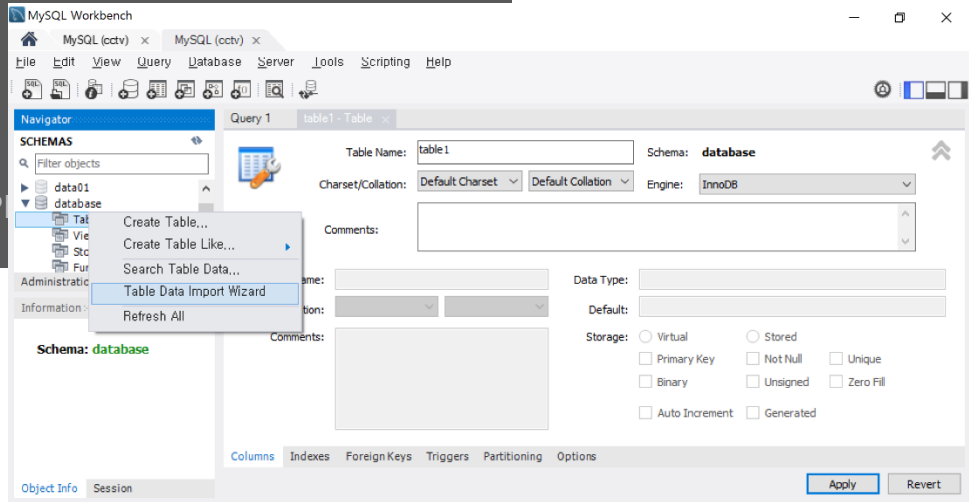


‘Apply’를 클릭하면 MySQL Workbench 가 다음과 같은 SQL 쿼리문을 작성해 줍니다. 마우스 클릭으로 지시한 명령을 코드로 변환하였습니다.



```
CREATE TABLE `database`.`table1`
(`id` INT NOT NULL
AUTO_INCREMENT,
```

- 데이터 로딩하기 :



IMPORT EXCEL, CSV Data

EXCEL 다운로드

안전비상벨위자정보

납시터정보

비산먼지발생사업정보

민방위대피시설정보

서울특별시

시군구(전체)

데이터기준일자 검색시작일 ~ 데이터기준일자 검색종료일

검색

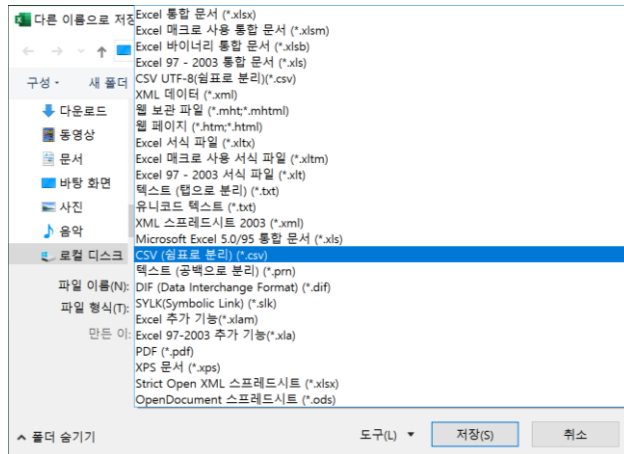
총 43,308 건

EXCEL

| 번호 | 설치목적구분 | 주소 | 설치연월 | 관리기관명 | 데이터기준일자 |
|----|--------|---|------|-------------|------------|
| 1 | 생활방범 | 서울특별시 동대문구청 용두동39-1017(답십리굴다리지하차도1) 신답초 | - | 서울특별시 동대문구청 | 2024-01-01 |
| 2 | 생활방범 | 서울특별시 동대문구청 고산자로 402 | - | 서울특별시 동대문구청 | 2024-01-01 |
| 3 | 생활방범 | 서울특별시 동대문구청 천정산로7길 30 | - | 서울특별시 동대문구청 | 2024-01-01 |
| 4 | 생활방범 | 서울특별시 동대문구청 천정산로7길 64 | - | 서울특별시 동대문구청 | 2024-01-01 |
| 5 | 생활방범 | 서울특별시 동대문구청 이문로9가길 11 | - | 서울특별시 동대문구청 | 2024-01-01 |
| 6 | 교통단속 | 서울특별시 동대문구청 이문로9길 52 | - | 서울특별시 동대문구청 | 2024-01-01 |
| 7 | 생활방범 | 서울특별시 동대문구청 이문로46길 68 | - | 서울특별시 동대문구청 | 2024-01-01 |

parking_2024.xls로 파일명 변경

엑셀에서 parking_2024.xls 파일을 불러온 후,
parking_2024.csv 다른이름으로 저장합니다.

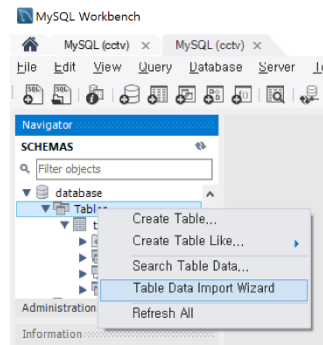


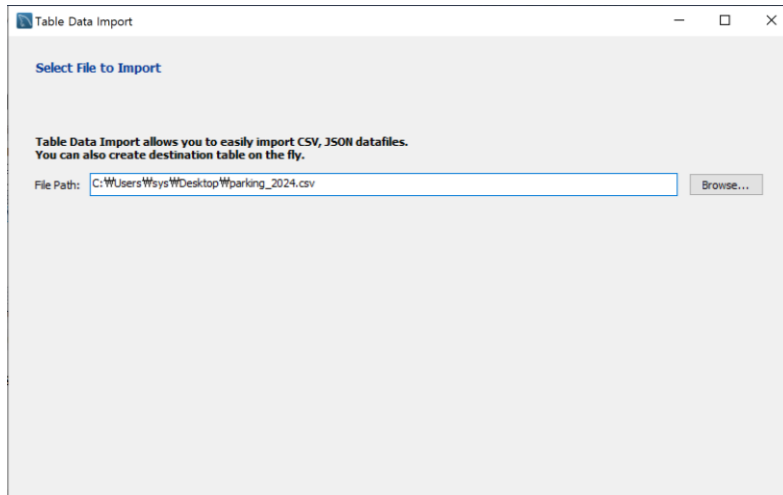
단, 여기서 확장자를 선택할 때 .csv(쉼표로 분리)를 선택합니다.

(CSV UTF-8 이 아님)

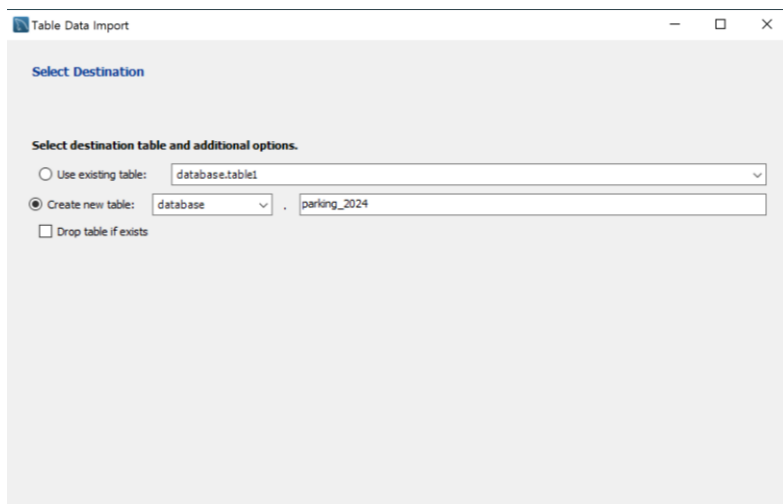
이제 .csv 파일을 MySQL Workbench에서 불러옵니다.

DB이름 위에서 마우스 우클릭한 후, 'Table Data Import Wizard'를 클릭합니다.

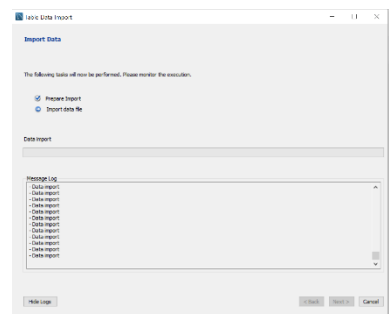




Parking_2024.csv파일을 클릭한 후,

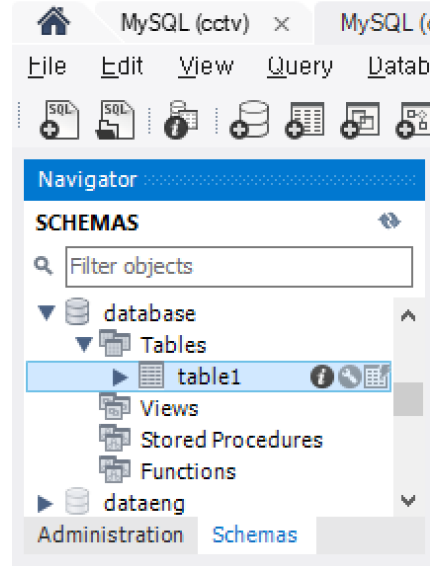


파일 전체를 DB(schema) 내부의 table로 불러오기(import)합니다.



데이터 수동으로 입력하기
(INSERT)

MySQL Workbench

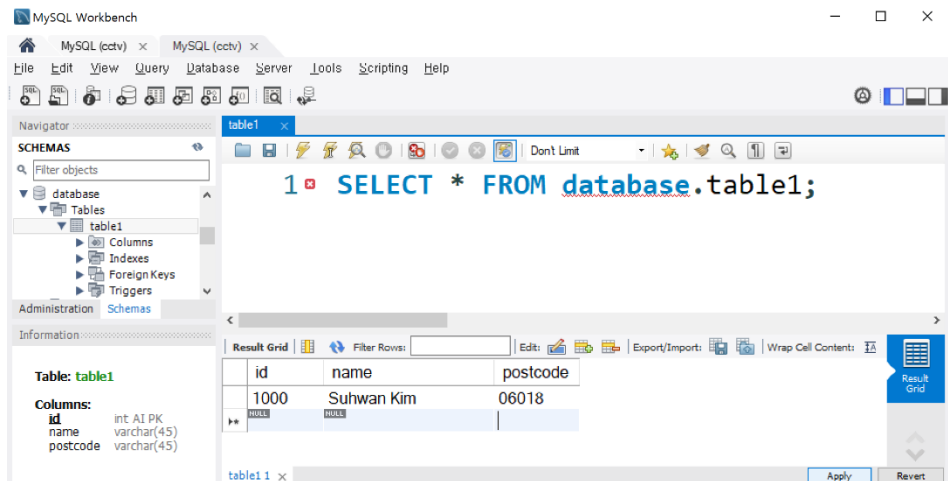


SELECT * FROM database.table1;

을 입력하고 실행한 후

하단 빈칸에 데이터를 입력,

‘Apply’를 클릭합니다.



‘Apply’를 클릭하면, 다음과 같은 SQL 쿼리문이 자동으로 실행됩니다.

```
INSERT INTO `database`.`table1` (`id`,  
`name`, `postcode`)  
VALUES ('1000', 'Suhwan Kim', '06018');
```

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Review the SQL Script to be Applied on the Database

1

2

3

4

INSERT INTO `database`.`table1` (`id`,
`name`, `postcode`)
VALUES ('1000', 'Suhwan Kim', '06018');

Apply SQL Script to Database

Review SQL Script

Apply SQL Script

Applying SQL script to the database

The following tasks will now be executed. Please monitor the execution.
Press Show Logs to see the execution logs.

Execute SQL Statements

SQL script was successfully applied to the database.

Message Log


Executing:
INSERT INTO `database`.`table1` (`id`, `name`, `postcode`) VALUES ('1000', 'Suhwan Kim', '06018');

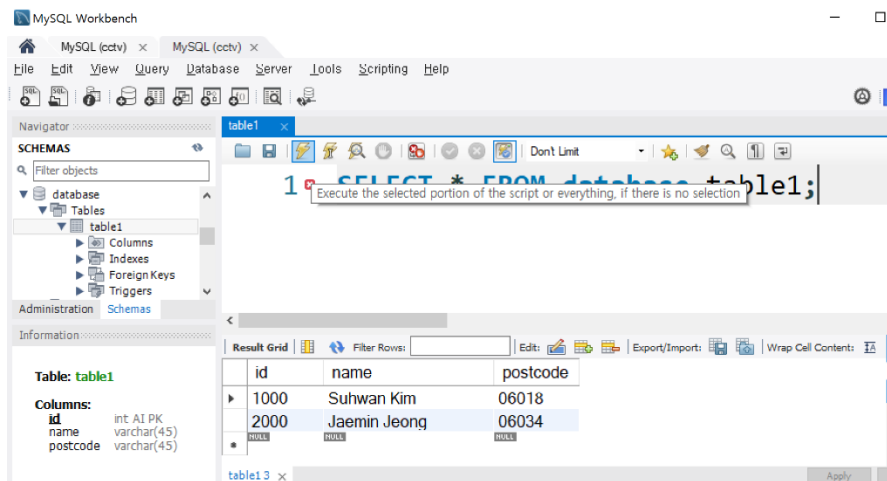
SQL script was successfully applied to the database.

20

2. 데이터 단순조회하기 (SELECT)

- `SELECT * FROM database.table1;`

쿼리문을 입력하여 실행해 봅니다. 상단  번개모양 버튼 또는
CTRL + ENTER.

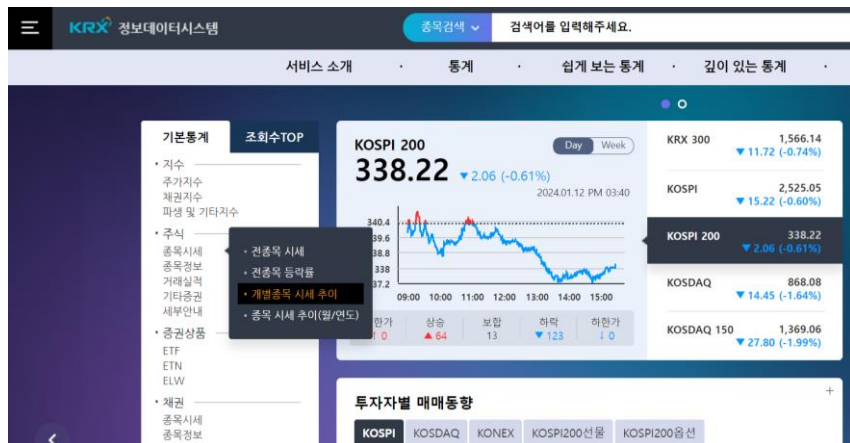


3. 데이터 조건조회하기 (SELECT)

(10 페이지) 삼성전자 주가 데이터를 활용합니다.
삼성전자 주식 데이터 다운로드

<http://data.krx.co.kr>

종목시세 -> 개별종목 시세 추이

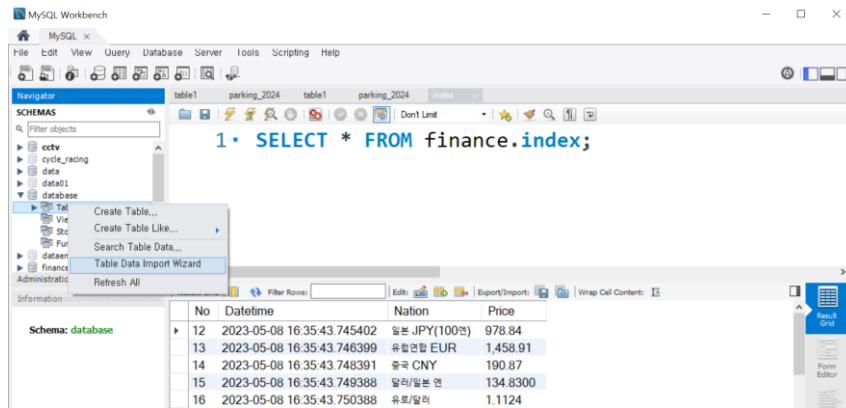


The screenshot shows the 'Individual Stock Price History' page for Samsung Electronics (005930). The page displays a table of daily stock prices from 2020.01.01 to 2024.01.12. The table includes columns for date, opening price, high, low, closing price, volume, and other metrics. The data shows a general upward trend in the stock price over the period.

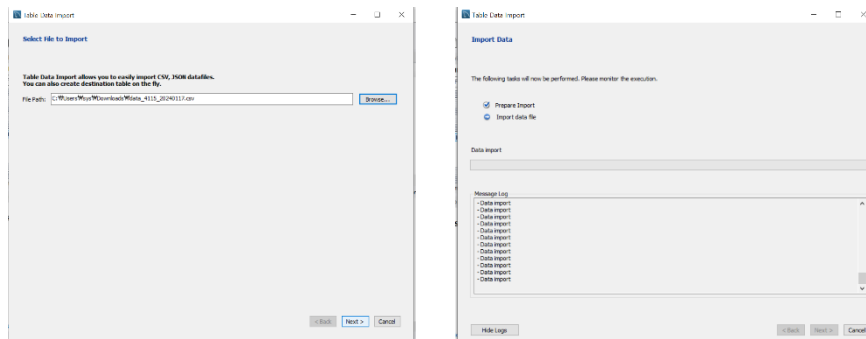
| 일자 | 종가 | 고가 | 저가 | 거래량 | 거래대금 | 시가총액 |
|------------|--------|--------|--------|------------|-------------------|----------------|
| 2024/01/12 | 73.100 | 73.100 | 72.800 | 12,954,138 | 950,604,343.500 | 436,391,104.40 |
| 2024/01/11 | 73.200 | 73.200 | 72.700 | 57,691,266 | 4,206,812,650.490 | 436,988,082.66 |
| 2024/01/10 | 73.600 | 73.600 | 73.200 | 20,259,529 | 1,497,565,371.180 | 439,375,995.66 |
| 2024/01/09 | 74.700 | 74.700 | 74.300 | 26,019,249 | 1,971,736,955.597 | 445,942,756.46 |
| 2024/01/08 | 76.500 | 76.500 | 76.400 | 11,088,724 | 852,681,337.648 | 456,688,365.07 |
| 2024/01/05 | 76.600 | 76.600 | 76.400 | 11,304,316 | 866,836,486.100 | 457,285,343.31 |
| 2024/01/04 | 76.600 | 76.600 | 76.100 | 15,324,439 | 1,173,912,901.977 | 457,285,343.31 |

날짜 범위 설정 후, 다운로드 버튼 클릭하여 .csv로 다운로드합니다.

Table Data Import Wizard를 사용하여 데이터를 가져옵니다.



이미 존재하는 ‘database’ DB 안쪽에 ‘Table’을 우클릭하여 data.csv 파일을 불러옵니다.



SELECT 명령어를 사용하여 단순조회해 봅시다.

```
SELECT * FROM database.samsung;
```

The screenshot shows the MySQL Workbench interface. The query editor at the top contains the SQL statement: `1 SELECT * FROM database.samsung;`. Below the editor, the 'Result Grid' displays the query results. The table has 11 columns: '일자' (Date), '종가' (Closing Price), '대비' (Change), '종량' (Volume), '시가' (Opening Price), '고가' (High Price), '저가' (Low Price), '거래량' (Trading Volume), '거래대금' (Trading Value), '시가총액' (Market Capitalization), and '상장주식수' (Number of Shares Outstanding). The results are sorted by '일자' in descending order, showing data from 2024/01/17 down to 2024/01/02.

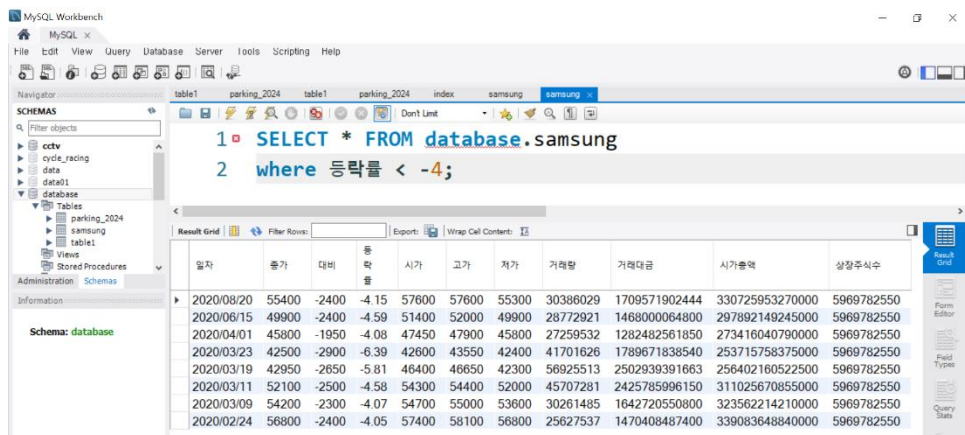
| 일자 | 종가 | 대비 | 종량 | 시가 | 고가 | 저가 | 거래량 | 거래대금 | 시가총액 | 상장주식수 |
|------------|-------|-------|-------|-------|-------|-------|----------|---------------|-----------------|------------|
| 2024/01/17 | 71900 | -700 | -0.96 | 73100 | 73300 | 71900 | 6966259 | 505891084200 | 429227365345000 | 5969782550 |
| 2024/01/16 | 72600 | -1300 | -1.76 | 73500 | 73700 | 72500 | 14760415 | 1075976773200 | 433406213130000 | 5969782550 |
| 2024/01/15 | 73900 | 800 | 1.09 | 73200 | 74000 | 73200 | 13212339 | 973180055524 | 441166930445000 | 5969782550 |
| 2024/01/12 | 73100 | -100 | -0.14 | 73000 | 74100 | 72800 | 13038939 | 956811865637 | 436391104405000 | 5969782550 |
| 2024/01/11 | 73200 | -400 | -0.54 | 72900 | 73600 | 72700 | 57691266 | 4206812650490 | 436988082660000 | 5969782550 |
| 2024/01/10 | 73600 | -1100 | -1.47 | 75000 | 75200 | 73200 | 20259529 | 1497565371180 | 439375995680000 | 5969782550 |
| 2024/01/09 | 74700 | -1800 | -2.35 | 77400 | 77700 | 74300 | 26019249 | 1971736955597 | 445942756485000 | 5969782550 |
| 2024/01/08 | 76500 | -100 | -0.13 | 77000 | 77500 | 76400 | 11088724 | 852681337648 | 456688365075000 | 5969782550 |
| 2024/01/05 | 76600 | 0 | 0 | 76700 | 77100 | 76400 | 11304316 | 866836486100 | 457285343330000 | 5969782550 |
| 2024/01/04 | 76600 | -400 | -0.52 | 76100 | 77300 | 76100 | 15324439 | 1173912901977 | 457285343330000 | 5969782550 |
| 2024/01/02 | 77000 | -2600 | -3.37 | 78500 | 78800 | 77000 | 21753844 | 1601800824924 | 450873256260000 | 5969782550 |

이번에는 SQL 쿼리문을 사용하여 ‘종가’ 컬럼의 값이 50000 미만인 값을 출력해 봅시다.

```
SELECT * FROM database.samsung  
where 종가 < 50000;
```


이번에는 등락률 < -4 인 값을 모두 검색해 봅시다. 명령문 끝에 문장이 완성되었다는 의미로 세미콜론(;)을 붙여줍니다.

```
SELECT * FROM database.samsung
where 등락률 < -4 ;
```



The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL query:

```
1 SELECT * FROM database.samsung
2 where 등락률 < -4;
```

The results are displayed in a table with the following columns and data:

| 일자 | 종가 | 대비 | 등락률 | 시가 | 고가 | 저가 | 거래량 | 거래대금 | 시가총액 | 상장주식수 |
|------------|-------|-------|-------|-------|-------|-------|----------|---------------|-----------------|------------|
| 2020/08/20 | 55400 | -2400 | -4.15 | 57600 | 57600 | 55300 | 30386029 | 1709571902444 | 330725953270000 | 5969782550 |
| 2020/06/15 | 49900 | -2400 | -4.59 | 51400 | 52000 | 49900 | 28772921 | 1469000064800 | 297892149245000 | 5969782550 |
| 2020/04/01 | 45800 | -1950 | -4.08 | 47450 | 47900 | 45800 | 27259532 | 1282482561850 | 273416040790000 | 5969782550 |
| 2020/03/23 | 42500 | -2900 | -6.39 | 42600 | 43550 | 42400 | 41701626 | 1789671838540 | 253715758375000 | 5969782550 |
| 2020/03/19 | 42950 | -2650 | -5.81 | 46400 | 46650 | 42300 | 56925513 | 2502939391663 | 256402160522500 | 5969782550 |
| 2020/03/11 | 52100 | -2500 | -4.58 | 54300 | 54400 | 52000 | 45707281 | 2425785996150 | 311025670855000 | 5969782550 |
| 2020/03/09 | 54200 | -2300 | -4.07 | 54700 | 55000 | 53600 | 30261485 | 1642720550800 | 323562214210000 | 5969782550 |
| 2020/02/24 | 56800 | -2400 | -4.05 | 57400 | 58100 | 56800 | 25627537 | 1470408487400 | 339083648840000 | 5969782550 |

이번에는 등락률 > 5 인 레코드를 모두 검색해 봅시다. 하루에 주가가 5% 이상 상승한 날짜 데이터를 모두 추려서 검색해 봅니다.

```
SELECT * FROM database.samsung
where 등락률 > 5;
```

The screenshot shows the MySQL Workbench interface. The query editor contains the following SQL query:

```
1 SELECT * FROM database.samsung
2 where 등록률 > 5;
```

The results are displayed in a grid with the following columns and data:

| 일자 | 종가 | 대 비 | 등록 률 | 시가 | 고가 | 저가 | 거래량 | 거래대금 | 시가총액 | 상장주식수 |
|------------|-------|--------|---------|-------|-------|-------|----------|---------------|-----------------|------------|
| 2023/09/01 | 71000 | 4100 | 6.13 | 66800 | 71000 | 66700 | 29738235 | 2064179447500 | 423854561050000 | 5969782550 |
| 2021/11/22 | 74900 | 3700 | 5.2 | 73300 | 75200 | 73000 | 27506623 | 2047227974600 | 447136712995000 | 5969782550 |
| 2021/01/08 | 88800 | 5900 | 7.12 | 83300 | 90000 | 83000 | 59013307 | 5083939899952 | 530116690440000 | 5969782550 |
| 2020/12/24 | 77800 | 3900 | 5.28 | 74100 | 78800 | 74000 | 32502870 | 2486986812800 | 464449082390000 | 5969782550 |
| 2020/07/28 | 58600 | 3000 | 5.4 | 57000 | 58800 | 56400 | 48431566 | 2803148376939 | 349829257430000 | 5969782550 |
| 2020/06/03 | 54500 | 3100 | 6.03 | 51800 | 55000 | 51700 | 49257814 | 2655435431458 | 325353148975000 | 5969782550 |
| 2020/03/24 | 46950 | 4450 | 10.47 | 43850 | 46950 | 43050 | 49801908 | 2254620196840 | 280281290722500 | 5969782550 |
| 2020/03/20 | 45400 | 2450 | 5.7 | 44150 | 45500 | 43550 | 49730008 | 2221272156100 | 271028127770000 | 5969782550 |

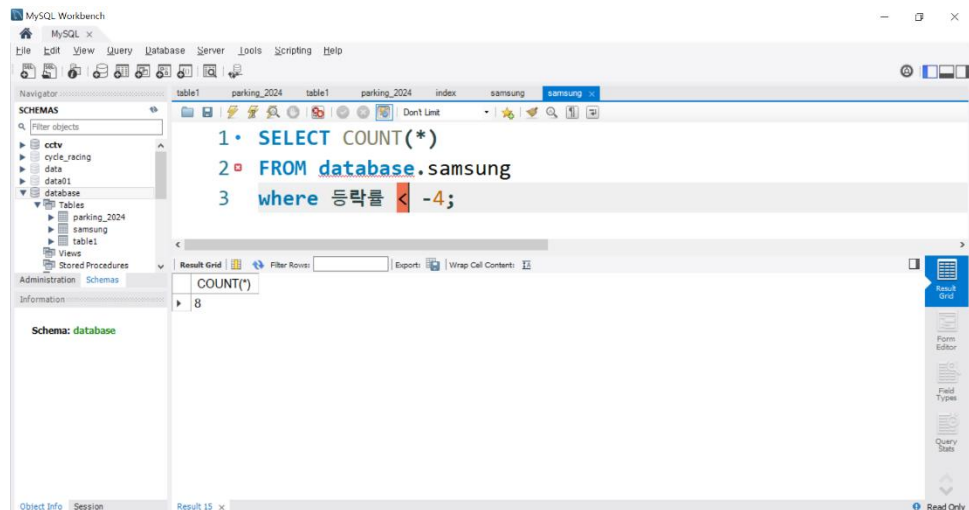
삼성전자 주가 데이터를 MySQL 로 가져와서, SELECT ~ WHERE 문으로 조건검색해 보았습니다.

5. 레코드의 개수를 세어 검색하기 (COUNT)

이번에는 이 SQL 쿼리문에 COUNT 함수를 이용하여, 레코드의 개수를 세어 봅시다.

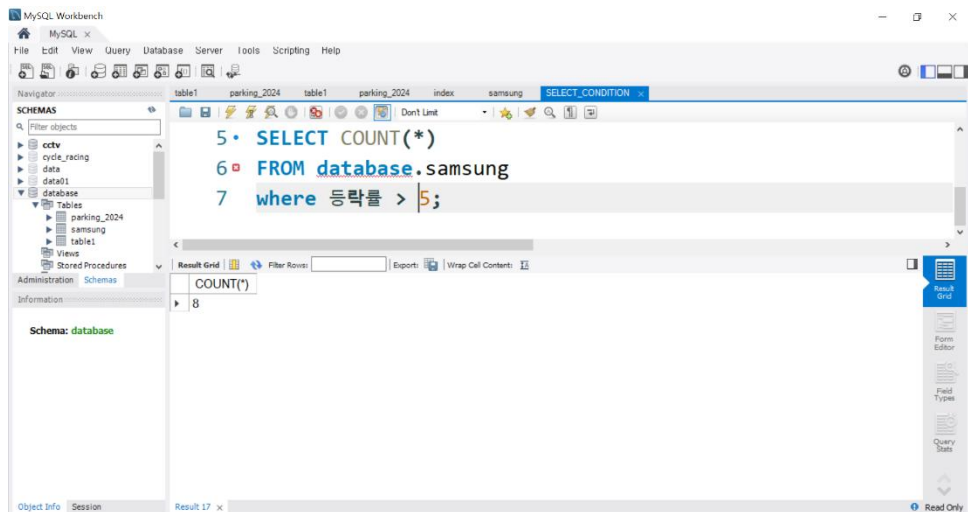
등락률 컬럼의 값이 -4 보다 작은 레코드의 개수를 세어 봅니다.

```
SELECT COUNT(*)  
FROM database.samsung  
where 등락률 < -4;
```



등락률 컬럼의 값이 5 보다 큰 레코드의 개수를 세어 봅니다

```
SELECT COUNT(*)  
FROM database.samsung  
where 등락률 > 5;
```



COUNT 함수를 사용하여 레코드의 개수를 세어 출력해 보았습니다.

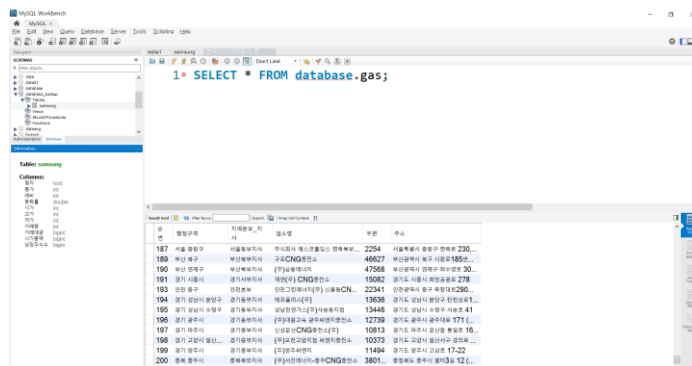
6. 데이터 추가/삭제/업데이트하기

공공데이터포털에서 다운받은 ‘도시가스충전소현황’을 불러옵니다.

<https://www.data.go.kr/data/15001508/fileData.do>

Data Import Wizard 실행 -> 데이터 가져오기 (‘database’ DB 내부에 테이블 이름은 ‘gas’로 설정하였습니다).

SELECT * FROM database.gas;



I. INSERT

INSERT문을 사용하여 한 줄을 추가해 보시다.

```
USE database;
```

```
INSERT INTO database.gas (순번, 행정구역, 지역본부_지사, 업소명, 우편, 주소)
```

```
VALUES ('201','경기 성남시 분당구','경기동부지사','에코플러스 (주)','13636','경기도 성남시 분당구 탄천상로 163번길 10');
```

SELECT * FROM database.gas; 를 실행하여 제대로 추가되었는지 확인합니다.

II. DELETE

DELETE문을 사용하여 한 줄을 삭제해 봅시다.

```
USE database;  
  
DELETE FROM database.gas WHERE (순번=190);  
  
DELETE FROM database.gas WHERE (순번=189);  
  
DELETE FROM database.gas WHERE (순번=188);  
  
SELECT * FROM database.gas;
```

III. UPDATE

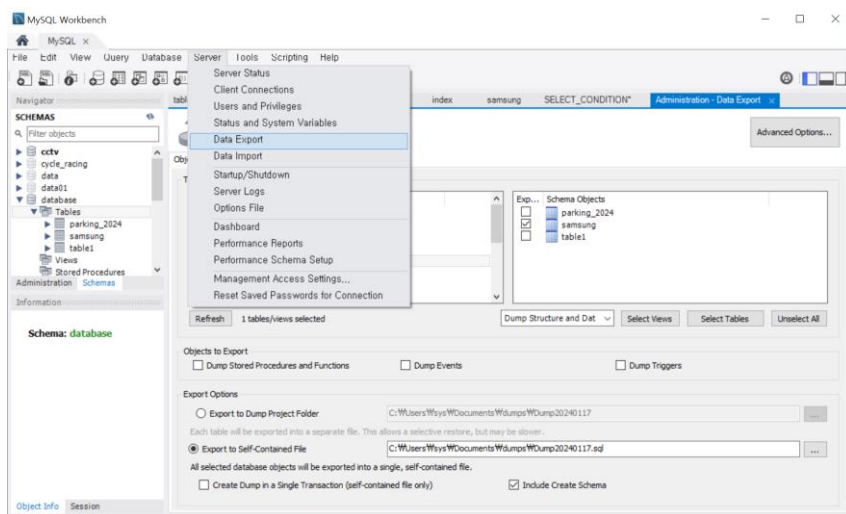
```
UPDATE database.gas SET 행정구역 = '경기 광주시' WHERE  
(순번 = 202);  
  
UPDATE database.gas SET 지역본부_지사 = '경기동부지사'  
WHERE (순번 = 202);  
  
UPDATE database.gas SET 업소명 = '(주)대원고속 광주씨엔지  
충전소' WHERE (순번 = 202);
```

7. 데이터 내보내기 / 가져오기

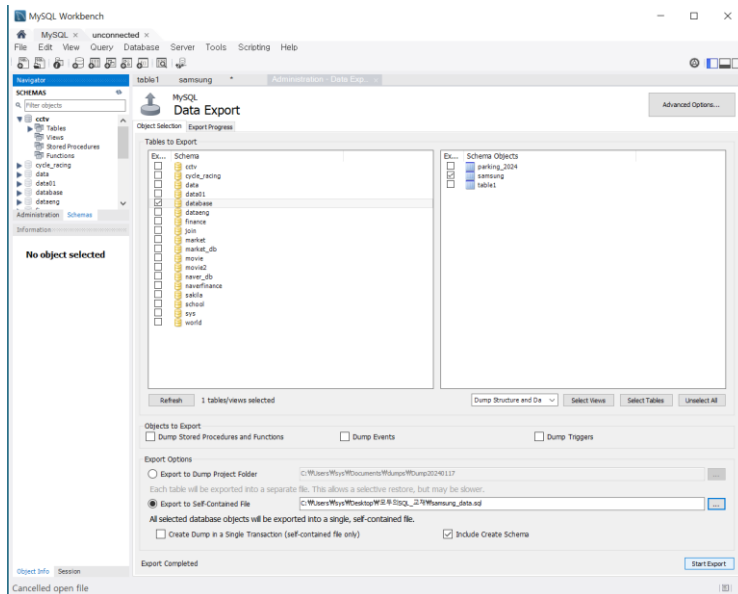
이번에는 Data Export를 사용하여 데이터 내보내기를 하여 봅시다.

SQL의 데이터는 매우 정확도가 높습니다. 데이터를 DB에서 DB로 옮길 때는, SQL 파일(.sql)로 내보내기하여 줍니다. 왜냐하면 이러한 방식이 .csv 파일로 데이터를 전송할 때보다 보다 정확하고 오류가 없기 때문입니다.

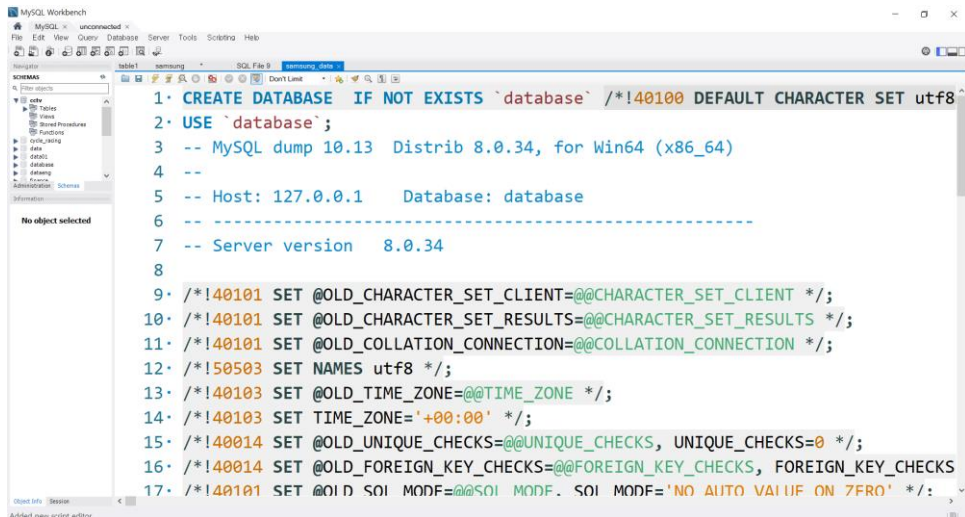
Server -> Data Export 메뉴를 클릭합니다.



파일명(samsung_data.sql)과 폴더를 지정하고, 'Start Export'를 누릅니다.



데이터를 export 한 뒤, 메뉴에서 Open SQL Script를 클릭하여
방금 저장한 samsung_data.sql 파일을 불러옵니다.



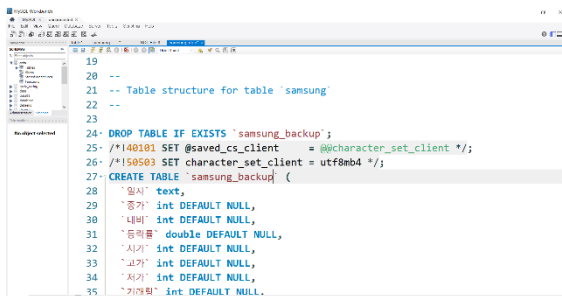
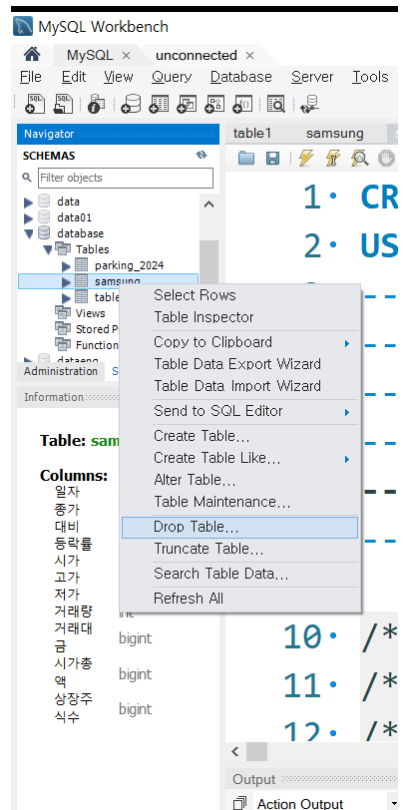
‘database’ DB(schema)의 ‘samsung’ 테이블의 내용이 이와 같이

sql 파일로 기록되어 있는 것을 알 수 있습니다. 이 파일을 실행하면, 자동으로 저장된 데이터가 DB로 입력됩니다.

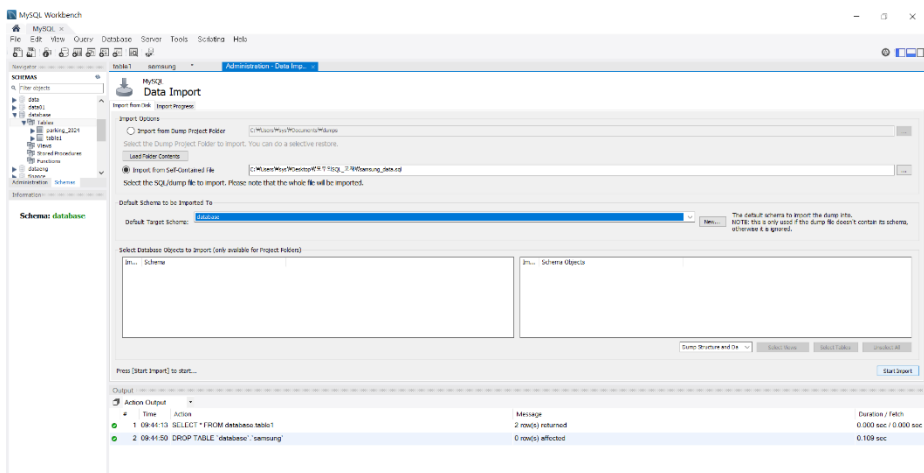
지금 있는 'samsung' 테이블을 삭제(drop)한 뒤, 다시 데이터 가져오기(import)를 해 봅시다.

Samsung (우클릭) -> Drop Table (테이블 삭제)

상단메뉴 Server -> Data Import -> .sql 데이터 파일선택



Reload 버튼을 눌러 데이터 Import가 잘 되었는지 확인합니다.



8. 뷰 생성하기 (CREATE VIEW)

지역본부_지사 = '경기동부지사'인 행만을 출력하는 SELECT문을 만들어 보시다.

```
SELECT 행정구역, 업소명, 주소  
FROM database.gas WHERE (지역본부_지사 = '경기동부지사');
```

이러한 SELECT문을 고정으로 두고 매번 불러와서 사용하고 싶다면, 아래와 같이 'VIEW'를 생성하면 됩니다. 'CREATE VIEW'문을 사용하여 위 SELECT문을 VIEW로 생성해 보시다.

```
CREATE VIEW database.수도권 AS  
SELECT 행정구역, 업소명, 주소  
FROM database.gas WHERE (지역본부_지사 = '경기동부지사');
```

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows the 'database' schema with a new view named '수도권' (Seodokwon) created. The main editor shows the SQL script for creating the view and a SELECT query to view its contents. The 'Result Grid' at the bottom displays the data returned by the SELECT query.

```
1 SELECT 행정구역, 업소명, 주소  
2 FROM database.gas WHERE (지역본부_지사 =  
3  
4 CREATE VIEW database.수도권 AS  
5 SELECT 행정구역, 업소명, 주소  
6 FROM database.gas WHERE (지역본부_지사 =  
7  
8 SELECT * FROM database.수도권;
```

| 행정구역 | 업소명 | 주소 |
|------------|--------------------|-----------------------|
| 경기 이천시 | (주)대원고속 이천공업지대C... | 경기도 이천시 이성대로 981-... |
| 경기 하남시 | 하남CNG충전소 | 경기도 하남시 하남대로284번... |
| 경기 이천시 | 부발CNG충전소 | 경기도 이천시 부발읍 중부대로1... |
| 경기 하남시 | 하남BRT CNG충전소 | 경기도 하남시 함주로 146 |
| 경기 성남시 분당구 | 성남천원가스(주) | 경기도 성남시 분당구 판교로 77... |
| 경기 성남시 분당구 | (주)경기고속분당CNG충전소 | 경기도 성남시 분당구 성남대로4... |
| 경기 용인시 처인구 | (주)상천리예비밴드CNG충전소 | 경기도 용인시 처인구 포곡읍 곡... |
| 경기 용인시 처인구 | (주)대원고속 용인CNG충전소 | 경기도 용인시 처인구 포곡읍 성... |

9. 오름차순/내림차순으로 정렬하기

```
SELECT * FROM database.gas  
  
WHERE (지역본부_지사 = '경기동부지사')  
  
ORDER BY 순번 DESC; # 내림차순 : 큰것부터 작은 것까지
```

```
SELECT * FROM database.gas  
  
WHERE (지역본부_지사 = '경기동부지사')  
  
ORDER BY 순번 ASC; #오름차순: 작은 것부터 큰 것까지
```

10. 그룹 별로 묶어 정렬하기 (Group By + Order BY)

11. 데이터를 그룹으로 묶어 평균 구하기 (AVG)

12. VIEW 생성하기

IV. JOIN문

- JOIN 문 - 두 개의 테이블을 서로 엮어서 정보를 추출하는 명령어
- 내부 JOIN (INNER JOIN) - 공통되는 부분만 추출
- 외부 JOIN (OUTER JOIN) - 모든 테이블을 병합
- 왼쪽 테이블 우선 JOIN (LEFT JOIN)
- 오른쪽 테이블 우선 JOIN (RIGHT JOIN)
- 상호 조인 (CROSS JOIN)

1. 내부 조인 (INNER JOIN)

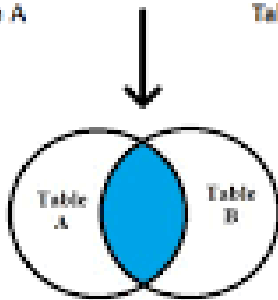
INNER JOIN = 양쪽 테이블 모두에서 일치하는 레코드를 반

| Student ID | Name |
|------------|------|
| 1001 | A |
| 1002 | B |
| 1003 | C |
| 1004 | D |

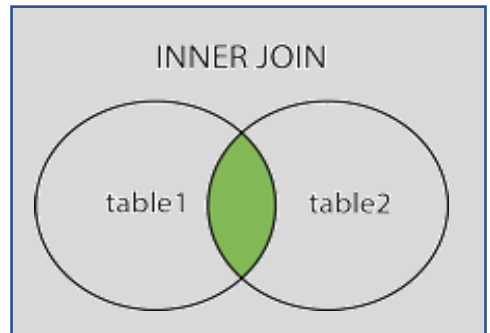
| Student ID | Department |
|------------|------------------|
| 1004 | Mathematics |
| 1005 | Mathematics |
| 1006 | History |
| 1007 | Physics |
| 1008 | Computer Science |

Table A

Table B



| Student ID | Name | Department |
|------------|------|-------------|
| 1004 | D | Mathematics |



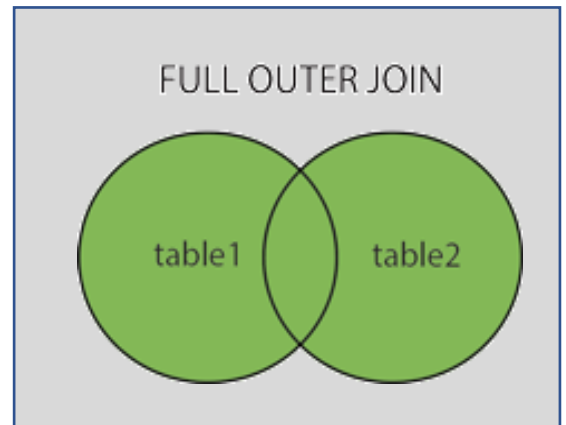
```
SELECT studentID, Name, Department
FROM table1
INNER JOIN table2 ON table1.studentID =
table2.studentID;
```

2. 외부 조인 (FULL OUTER JOIN)

| OrderID | CustomerID | EmployeeID | OrderDate | ShipperID |
|---------|------------|------------|------------|-----------|
| 10308 | 2 | 7 | 1996-09-18 | 3 |
| 10309 | 37 | 3 | 1996-09-19 | 1 |
| 10310 | 77 | 8 | 1996-09-20 | 2 |

| CustomerID | CustomerName | ContactName | Address | City | PostalCode | Country |
|------------|------------------------------------|----------------|-------------------------------|-------------|------------|---------|
| 1 | Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin | 12209 | Germany |
| 2 | Ana Trujillo Emparedados y helados | Ana Trujillo | Avda. de la Constitución 2222 | México D.F. | 05021 | Mexico |
| 3 | Antonio Moreno Taquería | Antonio Moreno | Mataderos 2312 | México D.F. | 05023 | Mexico |

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name
= table2.column_name;
```



1. 내부 조인(INNER JOIN)은 두 테이블에 모두 데이터가 있어야 결과가 나오는 반면,
2. 외부 조인(OUTER JOIN)은 한쪽에만 데이터가 있어도 결과를 출력합니다.

3. 왼쪽 외부 조인 (LEFT OUTER JOIN)

```
SELECT * FROM topic;
```

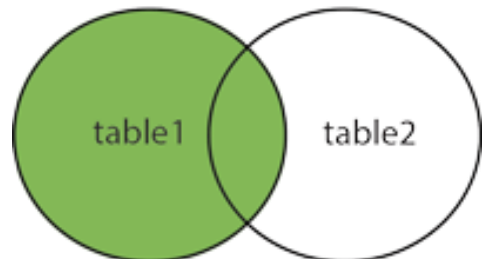
| tid | title | description | author_id |
|-----|------------|------------------|-----------|
| 1 | HTML | HTML is ... | 1 |
| 2 | CSS | CSS is ... | 2 |
| 3 | JavaScript | JavaScript is .. | 1 |
| 4 | Database | Database is ... | NULL |

```
SELECT * FROM author;
```

| aid | name | city | profile_id |
|-----|----------|--------|------------|
| 1 | egoing | seoul | 1 |
| 2 | leezche | jeju | 2 |
| 3 | blackdew | namhae | 3 |

- LEFT OUTER JOIN = “왼쪽 테이블의 내용은 모두 출력되어야 한다.”

LEFT JOIN



```
SELECT * FROM topic
```

```
LEFT JOIN author
```

```
ON topic.author_id = author.aid;
```


LEFT OUTER JOIN

```
SELECT * FROM topic
```

```
LEFT JOIN author
```

```
ON topic.author_id = author.aid;
```

```
SELECT * FROM topic;
```

| tid | title | description | author_id |
|-----|------------|------------------|-----------|
| 1 | HTML | HTML is ... | 1 |
| 2 | CSS | CSS is ... | 2 |
| 3 | JavaScript | JavaScript is .. | 1 |
| 4 | Database | Database is ... | NULL |

```
SELECT * FROM author;
```

| aid | name | city | profile_id |
|-----|----------|--------|------------|
| 1 | egoing | seoul | 1 |
| 2 | leezche | jeju | 2 |
| 3 | blackdew | namhae | 3 |

```
SELECT * FROM topic LEFT JOIN author ON topic.author_id = author.aid;
```

| tid | title | description | author_id | aid | name | city | profile_id |
|-----|------------|------------------|-----------|------|---------|-------|------------|
| 1 | HTML | HTML is ... | 1 | 1 | egoing | seoul | 1 |
| 2 | CSS | CSS is ... | 2 | 2 | leezche | jeju | 2 |
| 3 | JavaScript | JavaScript is .. | 1 | 1 | egoing | seoul | 1 |
| 4 | Database | Database is ... | NULL | NULL | NULL | NULL | NULL |

4. 오른쪽 외부 조인(RIGHT OUTER JOIN)

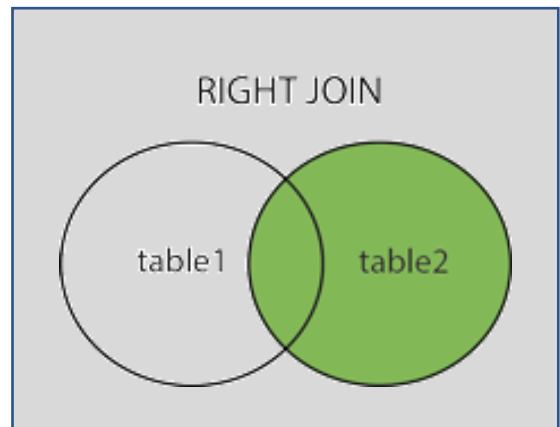
· orders 테이블

| order_id | customer_id | order_date |
|----------|-------------|------------|
| 1035 | 1 | 2021-10-14 |
| 1036 | 2 | 2021-10-15 |
| 1037 | 3 | 2021-10-16 |

· customer 테이블

| customer_id | name | address | phone |
|-------------|------|-----------|---------------|
| 1 | 김호준 | 경기도 파주시 | 010-0000-0002 |
| 2 | 이성민 | 서울특별시 송파구 | 010-0000-0101 |
| 3 | 남건우 | 강원도 춘천시 | 010-0000-0301 |
| 4 | 이성규 | 대전광역시 유성구 | 010-1234-5678 |

- RIGHT OUTER JOIN = “오른쪽 테이블의 내용은 모두 출력되어야 한다.”



```
SELECT order id, name, address, phone,
order_date
FROM orders

RIGHT JOIN customer

ON orders.customer.id = customer.customer.id
```

RIGHT OUTER JOIN

· orders 테이블

| order_id | customer_id | order_date |
|----------|-------------|------------|
| 1035 | 1 | 2021-10-14 |
| 1036 | 2 | 2021-10-15 |
| 1037 | 3 | 2021-10-16 |

· customer 테이블

| customer_id | name | address | phone |
|-------------|------|-----------|---------------|
| 1 | 김호준 | 경기도 파주시 | 010-0000-0002 |
| 2 | 이성민 | 서울특별시 송파구 | 010-0000-0101 |
| 3 | 남건우 | 강원도 춘천시 | 010-0000-0301 |
| 4 | 이성규 | 대전광역시 유성구 | 010-1234-5678 |

5. 상호 조인 (CROSS JOIN)



· 결과확인

| order_id | name | address | phone | order_date |
|----------|------|-----------|---------------|------------|
| 1035 | 김호준 | 경기도 파주시 | 010-0000-0002 | 2021-10-14 |
| 1036 | 이성민 | 서울특별시 송파구 | 010-0000-0101 | 2021-10-15 |
| 1037 | 남건우 | 강원도 춘천시 | 010-0000-0301 | 2021-10-16 |
| NULL | 이성규 | 대전광역시 유성구 | 010-1234-5678 | NULL |

```
SELECT order id, name, address, phone, order_date
FROM orders

RIGHT JOIN customer

ON orders.customer.id = customer.customer.id
```

조인 단원에서 다룬 내용

- 내부 JOIN (INNER JOIN) – 공통되는 부분만 추출
- 외부 JOIN (OUTER JOIN) – 모든 테이블을 병합
- 왼쪽 테이블 우선 JOIN (LEFT JOIN)
- 오른쪽 테이블 우선 JOIN (RIGHT JOIN)
- 상호 조인 (CROSS JOIN)