03

SQL 시작 전 준비 운동하기

03-1 DDL과 DML 빠르게 맛보기

- **데이터 조작 언어**(Data Manipulation Language, DML)는 테이블에서 데이터를 조회(SELECT), 삽입(INSERT), 수정(UPDATE), 삭제(DELETE)하는데 사용

- DML의 대상은 테이블이므로 DML을 사용하려면 반드시 테이블이 있어야 함. 테이블을 조작하는 언어를 **데이터 정의 언어**(Data Definition Language, DDL)라고 함

- DDL은 데이터베이스, 테이블, 뷰, 인덱스 등의 개체를 생성(CREATE), 삭제(DROP), 변경(ALTER)함

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터베이스 생성 및 삭제하기

▪ CREATE 문으로 데이터베이스 생성하기

- 데이터베이스를 생성하려면 CREATE 문을 사용하면 됨

Do it! MySQL 공식 문서에 소개된 CREATE 문

```
CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db_name
    [create_option] ...

create_option: [DEFAULT] {
    CHARACTER SET [=] charset_name
    | COLLATE [=] collation_name
    | ENCRYPTION [=] {'Y' | 'N'}
}
```

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터베이스 생성 및 삭제하기

▪ CREATE 문으로 데이터베이스 생성하기

- 이 문서는 명세를 포함되어 있어 매우 복잡해 보임
- 하지만 MySQL 실습에서 데이터베이스를 생성하는 쿼리는 다음과 같이 아주 간단함

데이터베이스를 생성하기 위한 CREATE 문의 기본 형식

```
CREATE DATABASE [데이터베이스 이름]
```

Do it! CREATE 문으로 데이터베이스 생성

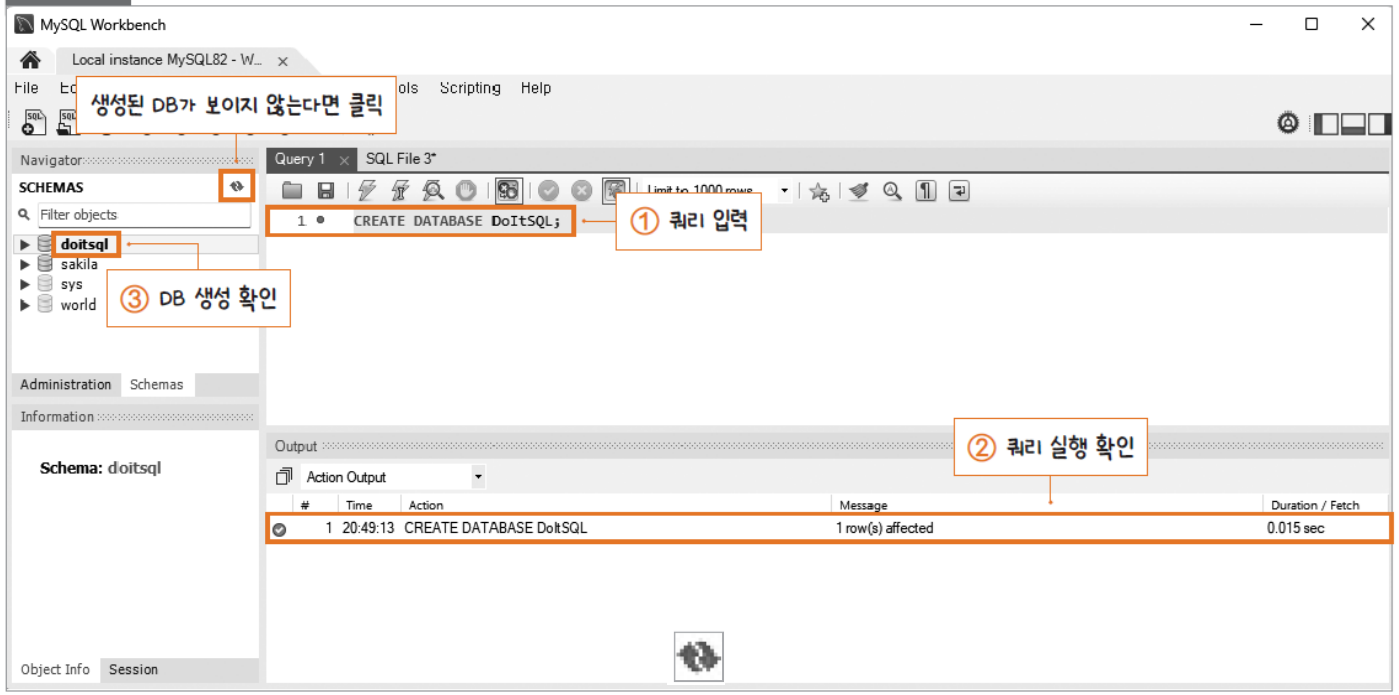
```
CREATE DATABASE DoItSQL;
```

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터베이스 생성 및 삭제하기

▪ CREATE 문으로 데이터베이스 생성하기

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 데이터베이스 생성 및 삭제하기

▪ DROP 문으로 데이터베이스 삭제하기

데이터베이스를 삭제하기 위한 DROP 문의 기본 형식

```
DROP DATABASE [데이터베이스 이름]
```

Do it! DROP 문으로 데이터베이스 삭제

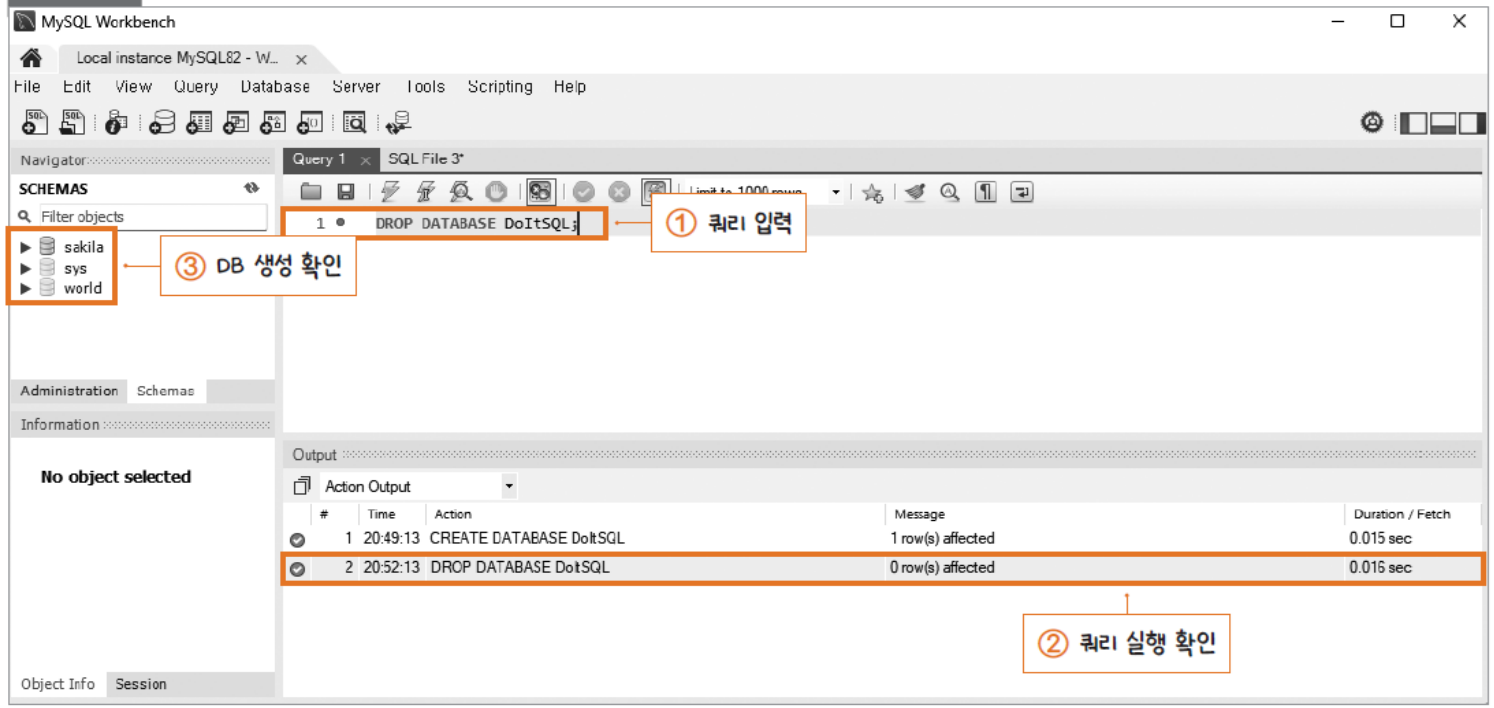
```
DROP DATABASE doitsql;
```

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터베이스 생성 및 삭제하기

▪ DROP 문으로 데이터베이스 삭제하기

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

- 데이터베이스를 생성 및 삭제할 때와 마찬가지로
  테이블을 생성할 때도 CREATE와 DROP문을 사용함

- 데이터베이스를 생성 또는 삭제할 때와 차이점이 있다면
  테이블을 생성할 때는 반드시 테이블이 위치할 데이터베이스를 먼저 선택해야 한다

03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

- 우리가 사용하는 MySQL이란 DBMS에 ‘doitsql’이라는 DB와 ‘sakila’라는 DB가 있다고 가정하자. DB에는 그림과 같이 여러 개의 테이블이 존재함
- 이와 같이 테이블은 데이터베이스 내부에 존재해야 하므로 반드시 테이블이 위치할 데이터베이스를 선택해야 함

03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

▪ CREATE 문으로 테이블 생성하기

1. 앞서 실습하면서 DB를 삭제했으므로 다시 CREATE 문으로 doitsql 데이터베이스를 생성하자.

   여러 작업을 하다 보면 데이터베이스의 위치가 변경될 수 있음.

   여기서는 doitsql 데이터베이스에서 작업하므로
   [Schemas] 탭에서 데이터베이스 이름을 더블클릭하거나
   USE 문으로 doitsql 데이터베스를 선택하자

03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

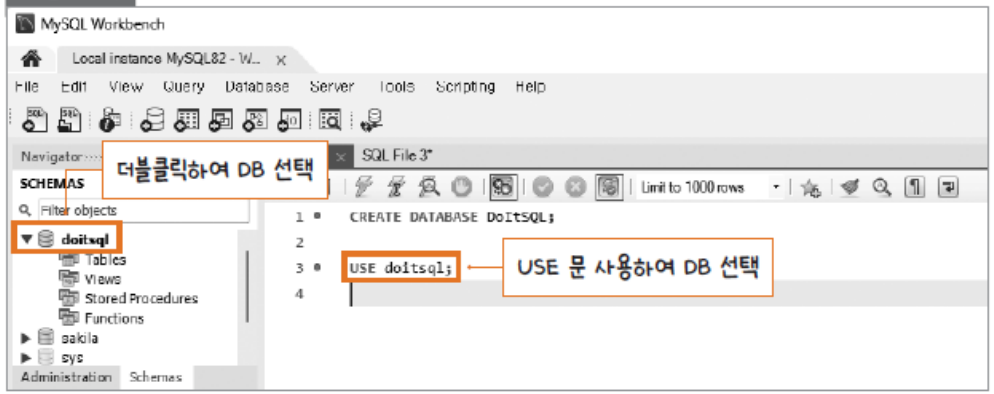
▪ CREATE 문으로 테이블 생성하기

1.

Do it! 데이터베이스 생성 후 선택

```
CREATE DATABASE doitsql;
USE doitsql;
```

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

▪ CREATE 문으로 테이블 생성하기

2. MySQL 공식 문서의 테이블 생성 관련 문법은 옵션이 매우 복잡함.
하지만 그 옵션들을 다 사용하지 않으므로
테이블 생성 쿼리는 다음과 같이 간단하게 입력하자.
이때, 열 이름은 테이블 안에서 고유해야 함

테이블을 생성하기 위한 CREATE 문의 기본 형식

```
CREATE TABLE 테이블 이름 (
[열 이름1 데이터 유형],
[열 이름2 데이터 유형],
(... 생략 ...)
)
```

03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

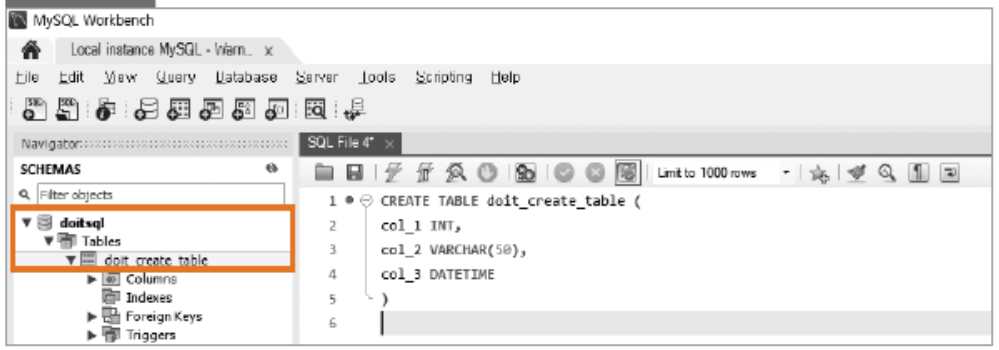
▪ CREATE 문으로 테이블 생성하기

3. doit_create_table 테이블을 생성해 보자.
   열은 col_1~col_3 이렇게 3개를 생성하며 각 열의 데이터 유형은 숫자, 문자, 날짜형임

Do it! CREATE 문으로 테이블 생성

```
CREATE                          TABLE
doit_create_table (
col_1 INT,
col_2 VARCHAR(50),
col_3 DATETIME
);
```

실행 결과



실행 후 [Schemas] 탭에서 [doitsql → Tables]를 클릭해 doit_create_table이 생성되었는지 확인해 보자

03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

▪ DROP 문으로 테이블 삭제하기

- 데이터베이스를 삭제할 때와 마찬가지로 테이블 삭제는 DROP 문을 사용하면 됨. 이때, 테이블 삭제도 데이터베이스를 삭제할 때처럼 즉시 사라지므로 주의해야 함

Do it! DROP 문으로 테이블 삭제
DROP TABLE doit_create_table;

실행 결과

03-1 DDL과 DML 빠르게 맛보기

▪ 테이블 생성 및 삭제하기

▪ DROP 문으로 테이블 삭제하기

- 가끔 테이블이 삭제되지 않는 경우도 있음

- 현재 삭제하려는 테이블이 다른 테이블과 종속 관계에 있으며,
  상위 테이블일 때 삭제할 수 없음
- 종속 관계에서 상위 테이블을 삭제하고 싶다면
  하위 테이블과의 종속 관계를 제거하고 하위 테이블을 모두 삭제해야 함
- 실습에서 아직 종속 관계를 지정한 적이 없으므로
  현재는 무리 없이 삭제가 실행될 것임

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ INSERT 문으로 데이터 삽입하기

- 데이터를 삽입할 때는 INSERT 문을 사용함

테이블을 생성하기 위한 INSERT 문의 기본 형식

INSERT INTO 테이블 이름 ([열1, 열2, ...]) (VALUES [값1, 값2, ...])

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ INSERT 문으로 데이터 삽입하기

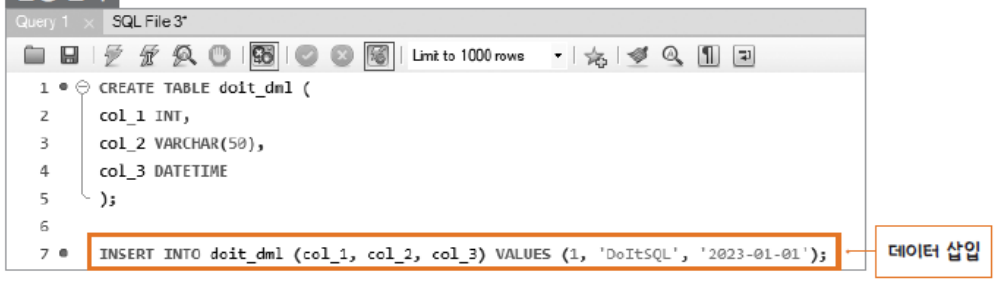
1. 데이터를 삽입하기에 앞서 먼저 테이블을 생성해 보자.
   doit_dml이라는 테이블을 생성한 후, 각 열에 데이터를 입력하고
   마지막에 데이터를 삽입해 보도록 함

Do it! INSERT 문으로 데이터 삽입

```
CREATE TABLE doit_dml (
col_1 INT,
col_2 VARCHAR(50),
col_3 DATETIME
);

INSERT INTO doit_dml (col_1, col_2, col_3) VALUES (1, 'DoItSQL', '2023-01-01');
```

실행 결과



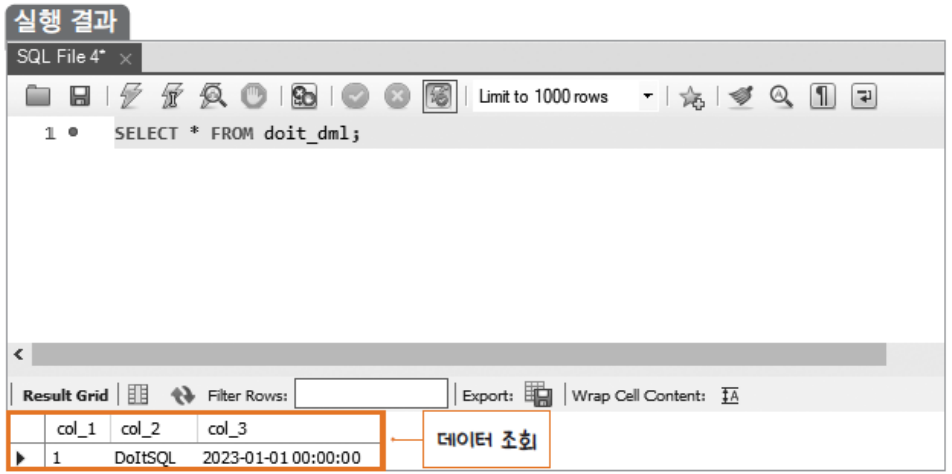
03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ INSERT 문으로 데이터 삽입하기

2. 쿼리를 실행한 후, doit_dml 테이블에서 방금 삽입한 데이터를 확인해 보자.
   이때, 다음과 같이 SELECT 문을 사용하면 데이터를 조회할 수 있음

Do it! 테이블 조회하여 삽입한 데이터 확인
SELECT * FROM doit_dml



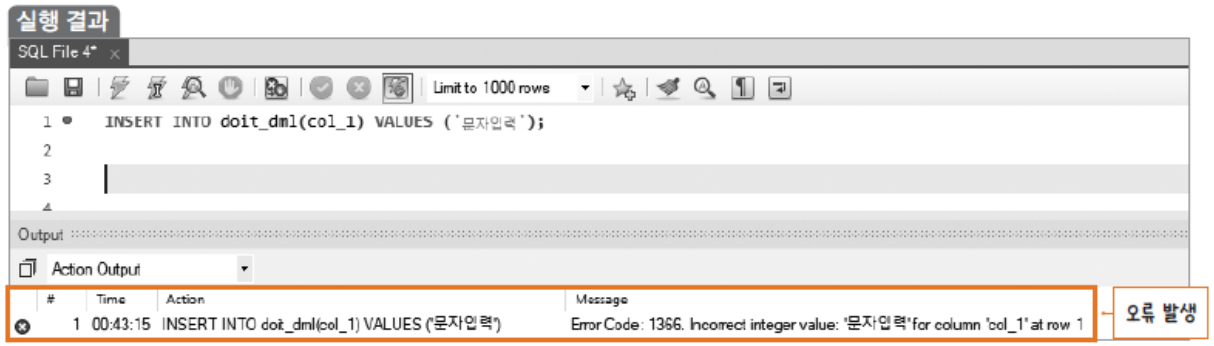
▪ 데이터 삽입, 수정, 삭제하기

▪ INSERT 문으로 데이터 삽입하기

3. 만약 데이터를 삽입할 때 앞서 지정한 데이터 유형과 맞지 않으면 오류가 발생함. 다음은 숫자형 열에 문자형을 삽입하여 오류가 발생한 예임

Do it! 데이터 유형 불일치로 인한 오류 발생 예
INSERT INTO doit_dml(col_1) VALUES ('문자 입력');

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ INSERT 문 더 알아보기

1. 테이블에 데이터를 삽입할 때 col_1, col_2와 같은 열 이름을 생략할 수 있음.

   열 이름을 생략하려면 다음과 같이 VALUES문 뒤에 테이블의 열 순서와 개수에 맞춰 데이터를 채워야 함

Do it! 열 이름 생략하고 데이터 삽입
INSERT INTO doit_dml VALUES (2, '열 이름 생략', '2023-01-02');

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

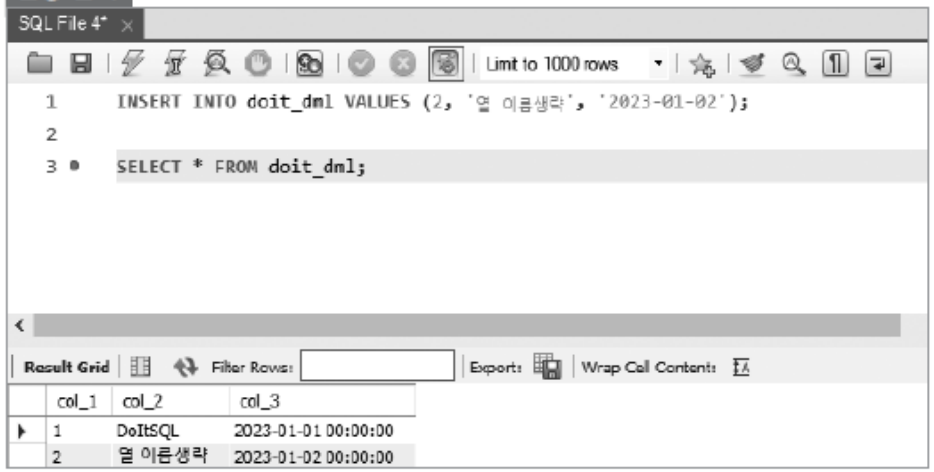
▪ INSERT 문 더 알아보기

1. 쿼리를 실행한 후 결과를 보면 열 이름을 생략했음에도 데이터가 잘 삽입된 것을 확인할 수 있음

Do it! 삽입된 데이터 확인

```
SELECT * FROM doit_dml;
```

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

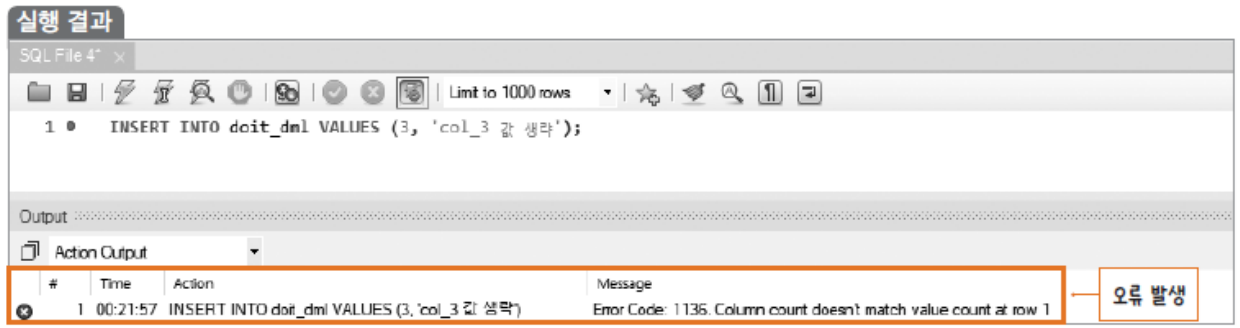
▪ INSERT 문 더 알아보기

2. 만약 col_3 위치에 해당하는 값을 입력하지 않으면
테이블의 열 개수와 입력한 값의 개수가 일치하지 않으므로 다음과 같이 오류가 발생함

Do it! 열 개수 불일치로 인한 오류 발생

```
INSERT INTO doit_dml VALUES (3, 'col_3 값 생략');
```

실행 결과



▪ 데이터 삽입, 수정, 삭제하기

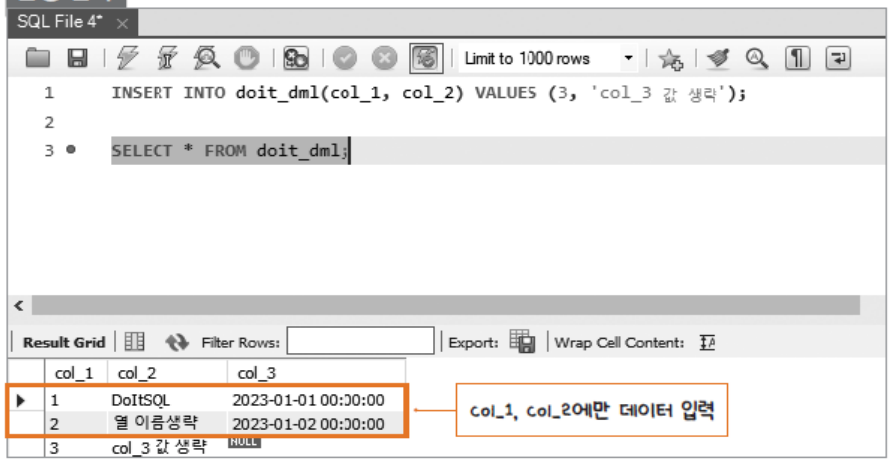
▪ INSERT 문 더 알아보기

3. col_1, col_2 열에만 데이터를 삽입하려면 테이블 이름 다음에 삽입하고자 하는 열만 소괄호 안에 나열하면 됨.

Do it! 특정 열에만 데이터 삽입

```
INSERT INTO doit_dml(col_1, col_2) VALUES (3, 'col_3 값 생략');
```

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

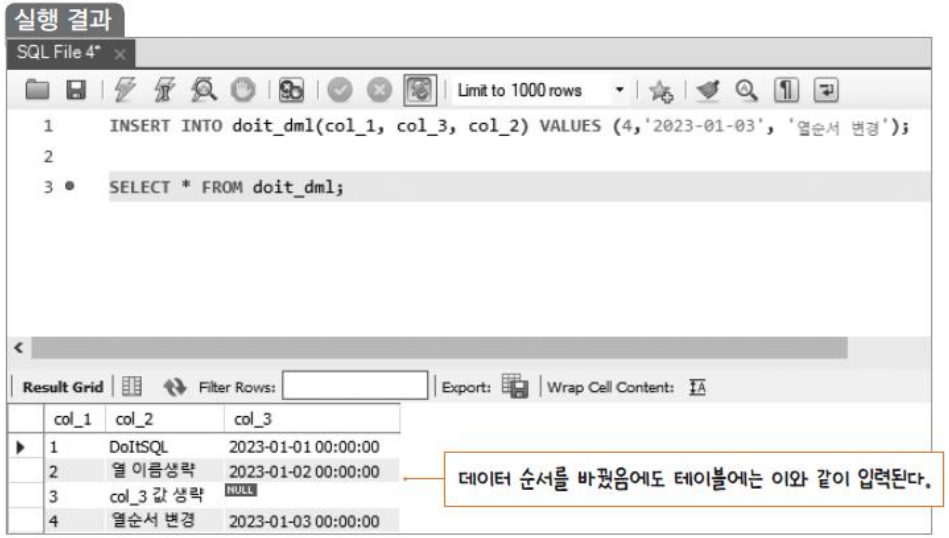
▪ INSERT 문 더 알아보기

4. 삽입하려는 데이터의 순서를 바꿀 수도 있음.
   대상 열과 삽입할 데이터를 맞춰 소괄호에 나열하면 됨

Do it! 삽입할 데이터의 순서 변경

```
INSERT INTO doit_dml(col_1, col_3, col_2)
VALUES (4,'2023-01-03', '열순서 변경');
```

실행 결과



03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ INSERT 문 더 알아보기

5. 여러 데이터를 한 번에 삽입하고 싶다면 INSERT 문을 여러 번 작성하면 되지만 삽입할 값을 소괄호로 묶어 쉼표로 구분하는 방법도 있음.

   이렇게 하나의 INSERT 문에 소괄호로 묶은 VALUES값을 사용하면 하나의 INSERT 구문으로 데이터를 여러 행 입력할 수 있어 사용자는 타이핑을 적게 한다는 장점도 있지만, 데이터베이스 관점에서 성능적으로도 유리함

Do it! 여러 데이터 한 번에 삽입

```
INSERT INTO doit_dml(col_1, col_2, col_3)
VALUES (5, '데이터 입력5', '2023-01-03'), (6, '데이터 입력6', '2023-01-03'),
(7, '데이터 입력7', '2023-01-03');
```

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ UPDATE 문으로 데이터 수정하기

- 이미 테이블에 삽입된 데이터를 수정하려면 UPDATE 문을 사용하면 됨.
  UPDATE 문의 기본 형식은 다음과 같음

UPDATE 문의 기본 형식

```
UPDATE 테이블 이름 SET [열1 = 값1, 열2 = 값2, ...]
WHERE [열 = 조건]
```

- UPDATE 문에서 WHERE 문을 생략할 수 있음.
  하지만 WHERE 문의 조건을 누락하면 테이블의 전체 데이터를 수정하므로 사용할 때 항상 주의해야 함

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ UPDATE 문으로 데이터 수정하기

1. 다음은 col_1의 값이 4인 행의 col_2 열의 값을 변경함.
그런데 쿼리를 실행하면 오류가 발생함. 문법은 틀리지 않았는데 오류가 발생한 이유는 무엇일까?

Do it! UPDATE 문으로 데이터 수정 1

```
UPDATE doit_dml SET col_2 = '데이터 수정'
WHERE col_1 = 4;
```

```
Error Code: 1175. You are using safe update mode and you tried to update a table without a WHERE that uses a KEY column.
To disable safe mode, toggle the option in Preferences -> SQL Editor and reconnect.
```

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ UPDATE 문으로 데이터 수정하기

1. 이 오류는 기본키가 없는 테이블에서 데이터를 수정할 때
WHERE 문에서 참고할 키 열이 없기 때문임.

   이와 같은 오류는 일종의 안전 모드가 작동된 셈임.
   이렇게 안전 모드가 작동하는 이유는 키 값을 사용하지 않고 데이터를 수정하거나
   삭제할 때 의도하지 않게 전체 데이터를 삭제하지 않도록 방지하기 위함임

2. 안전 모드를 비활성화하고 쿼리를 실행할 수 있음.

   안전 모드는 MySQL 워크벤치의 설정을 변경하여 비활성화할 수도 있고,
   쿼리를 사용하여 현재 접속되어 있는 세션에서만 비활성화를 적용할 수 있음.
   MySQL 워크벤치의 설정 자체를 변경해 보자.
   이 경우, MySQL 워크벤치를 사용할 때에도 이 설정이 유지됨

03-1 DDL과 DML 빠르게 맛보기

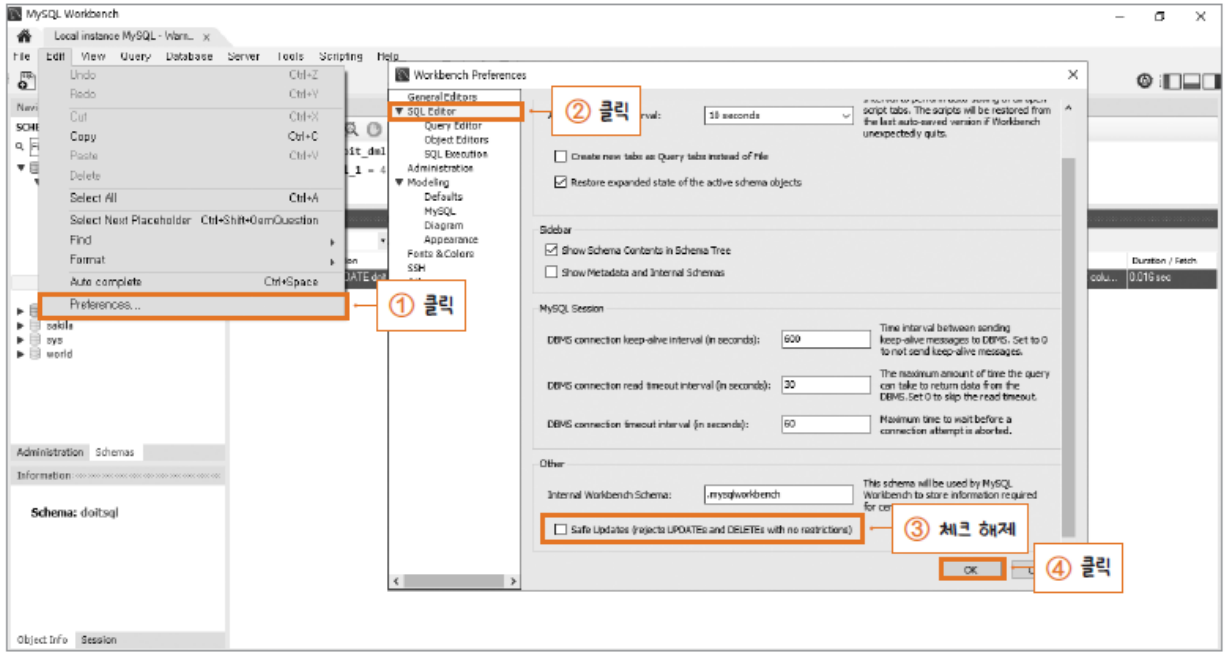
▪ 데이터 삽입, 수정, 삭제하기

▪ UPDATE 문으로 데이터 수정하기

2. 먼저, [Edit → Preference]를 클릭한 후,
Workbench Preferences 팝업 창이 나타나면 [SQL Editor]을 클릭함.

   그다음, 창 아래에 있는 [Safe Update(...)]의 체크 박스를 해제함.

   마지막으로 [OK]를 눌러 설정을 완료하고,
해당 옵션이 적용될 수 있도록 MySQL 워크벤치를 재실행함



▪ 데이터 삽입, 수정, 삭제하기

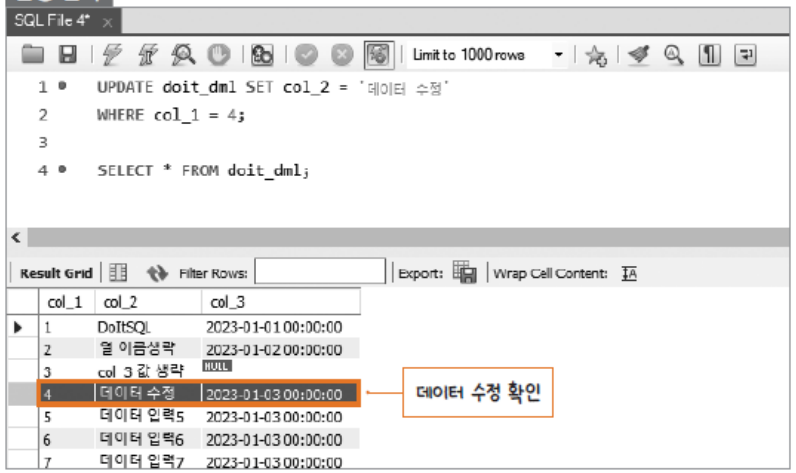
▪ UPDATE 문으로 데이터 수정하기

3. 안전 모드를 비활성화한 뒤, 이전에 오류가 났던 UPDATE 문을 다시 실행해 보자

Do it! UPDATE 문으로 데이터 수정 2

```
UPDATE doit_dml SET col_2 = '데이터 수정'
WHERE col_1 = 4;
```

실행 결과



03-1 DDL과 DML 빠르게 맛보기

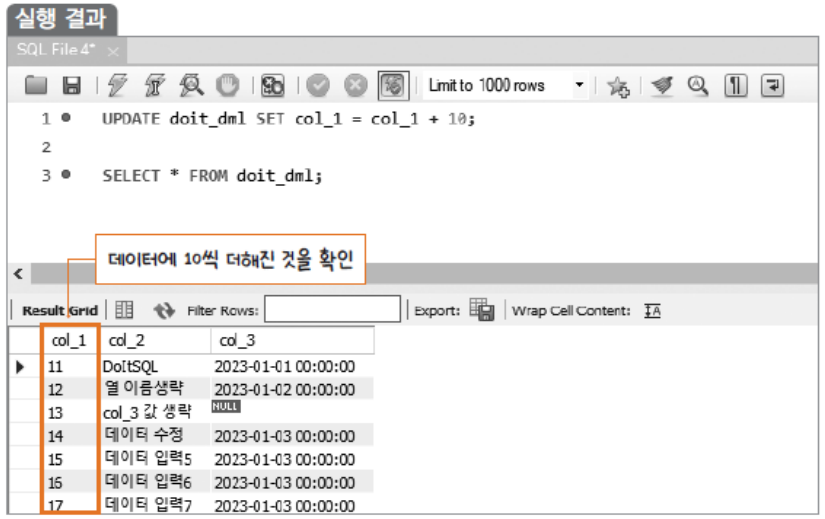
▪ 데이터 삽입, 수정, 삭제하기

▪ UPDATE 문으로 데이터 수정하기

4. 이번에는 col_1 열 전체에 10을 더하는 쿼리를 작성해 보자.
이 쿼리는 앞서 언급한 WHERE 문이 없고, 테이블의 전체 데이터에 영향을 주는 쿼리임

Do it! UPDATE 문으로 테이블 전체 데이터 수정

```
UPDATE doit_dml SET col_1 = col_1 + 10;
```



03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

▪ DELETE 문으로 데이터 삭제하기

- 입력된 데이터를 삭제하려면 DELETE 문을 사용함
- DELETE 문의 사용 방법은 UPDATE 문과 비슷함

DELETE 문의 기본 형식

```
DELETE FROM 테이블 이름 WHERE [열 = 조건]
```

- UPDATE 문과 동일하게 WHERE 문의 조건이 누락되면 전체 데이터를 삭제하므로 사용할 때 항상 주의해야 함

03-1 DDL과 DML 빠르게 맛보기

▪ 데이터 삽입, 수정, 삭제하기

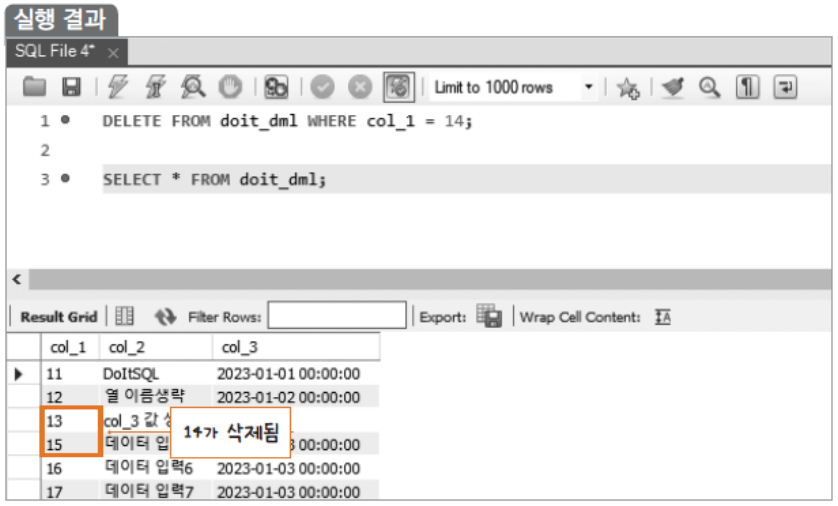
▪ DELETE 문으로 데이터 삭제하기

1. 다음은 col_1이 14인 데이터만 삭제하는 쿼리임

Do it! DELETE 문으로 데이터 수정

```
DELETE FROM doit_dml WHERE col_1 = 14;
```

실행 결과



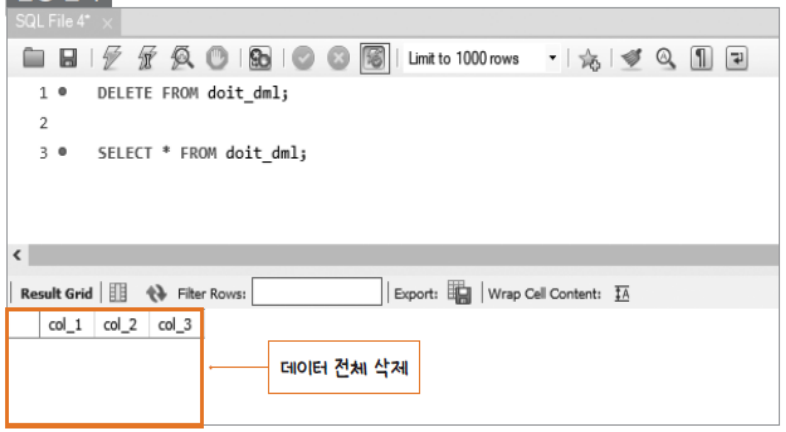
▪ 데이터 삽입, 수정, 삭제하기

▪ DELETE 문으로 데이터 삭제하기

2. 테이블에 있는 전체 데이터를 삭제하려면 WHERE 문 없이 쿼리를 실행하면 됨.
   다음 쿼리를 실행하면 테이블에 데이터가 하나도 남아 있지 않은 것을 확인할 수 있음

Do it! DELETE 문으로 테이블 전체 데이터 삭제
DELETE FROM doit_dml;

실행 결과



03-2 데이터베이스 모델링 이해하기

- 데이터베이스 모델링은 데이터베이스를 설계할 때
  효율적으로 데이터를 저장할 곳을 마련하기 위해 미리 설계하는 단계로,
  건축물에 빗대어 설명하면 건축 설계도를 만드는 것과 유사하다

- 설계도가 엉망인 상태로 시작해서 집을 짓는 동안 설계도를 변경하고 적용하는 일은
  쉬운 일이 아니다.
- 데이터베이스도 건축 설계와 마찬가지로
  잘못 만들어진 데이터베이스는 수정하기 힘들 뿐만 아니라 성능 또한 좋지 못하다

- 개발자들 사이에서는 이런 말을 할 정도이다

“DB 튜닝의 끝은 데이터베이스 모델링이다.”

03-2 데이터베이스 모델링 이해하기

▪ 데이터베이스 모델링의 개념과 필요성

- 데이터베이스 모델링은 수집한 정보와 관리 시스템을 시각적 표현하거나 청사진을 생성하는 과정으로,

  데이터 관리 시스템을 구축하기 위해
  어떤 데이터가 존재하는지 또는
  업무에 필요한 정보는 무엇인지를 분석하는 방법이기도 함

- 데이터베이스 모델링에서 좋지 못한 결과물이 만들어지면 실제 데이터베이스가 완성되었을 때의 결과물도 좋지 못함
- 시간이 지날수록 데이터베이스의 규모는 커지고 데이터의 양은 많아지는데, 잘못 설계하여 발생하는 다양한 성능 저하 문제나 확장 문제는 비즈니스 전체를 위험에 빠뜨릴 수 있음

03-2 데이터베이스 모델링 이해하기

▪ 데이터베이스 모델링의 개념과 필요성

▪ 데이터베이스 모델링의 필요성

데이터베이스 모델링의 필요성

- DBMS 구축에 필요한 다양한 기술을 효율적으로 적용하는 방안을 제시한다.
- 데이터베이스 설계 및 생성 속도와 효율성을 촉진시킨다.
- 조직의 데이터를 문서화하고 데이터 관련 시스템을 설계할 때 일관성을 조정한다.
- 업무 조직과 기술 조직 간의 의사소통을 원활히 하는 도구 또는 중재의 역할을 한다.

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

- 데이터 모델링은 **개념적 데이터 모델, 논리적 데이터 모델, 물리적 데이터 모델**로 나눌 수 있음
- 3가지 모델링 유형의 특징을 자동차 대리점을 예로 들어 살펴보자

- **개념적 데이터 모델**
  - 비즈니스 이해 관계자와 분석가가 개념적으로 모델을 생성하는 것으로, 공식 데이터를 활용해 모델링하는 것보다는 요구 사항을 도출하고 **프로젝트의 범위와 설계를 어떻게 할 것인지를** 정의하는 단계
  - 그러므로 간단한 다이어그램 정도로 결과물이 도출됨

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 개념적 데이터 모델

- 자동차 대리점을 만든다고 생각해 보면
  개념적 데이터 모델에서 도출해야 하는 항목은 다음과 같음

  - 대리점이 보유한 다양한 매장 정보를 나타내는 Showrooms 엔티티
  - 대리점이 현재 보유하고 있는 자동차 여러 대를 나타내는 Cars 엔티티
  - 대리점에서 자동차를 구매한 모든 고객을 나타내는 Customers 엔티티
  - 실제 판매 정보를 나타내는 Sales 엔티티
  - 대리점에서 일하는 모든 판매원 정보를 나타내는 Salesperson 엔티티

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 개념적 데이터 모델

- 이 개념적 데이터 모델에는 다음과 같은 비즈니스 요구 사항도 반드시 포함되어야 함
- 비즈니스 요구 사항은 사업이 지속되는 동안 끊임없이 변하기 때문에 처음부터 완벽하게 포함시킬 수는 없지만 최대한 도출하는 것이 좋음

- 모든 자동차는 특정 대리점 소속으로 운영된다.
- 모든 자동차에는 브랜드 이름과 제품 번호가 존재한다.
- 모든 판매에는 최소한 판매원 1명과 고객 1명이 연결된다.
- 모든 고객은 전화번호와 이메일 주소를 제공한다.

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 개념적 데이터 모델

- 이러한 방식으로 개념적 데이터 모델링을 위한 항목들을 도출할 수 있음
- 개념적 데이터 모델링은 다음과 같은 절차를 통해 이루어짐

개념적 데이터 모델링의 절차

주제 영역 도출 → 핵심 엔티티 도출 → 엔티티 간 관계 설정 → 엔티티 속성 정의 → 엔티티 식별자 정리

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 논리적 데이터 모델

- 개념적 데이터 모델에서 도출한 엔티티를 **기술적 데이터 구조와 연결하는** 단계로,
  개념적 데이터 모델에서 식별된 데이터와 복잡한 데이터 간 관계를 담고 있음
- 논리적 데이터 모델은 주로 데이터 아키텍트와 분석가 등이 협업을 통해 만듦
- 이러한 모델을 표현하고 생성하기 위해
  공식 데이터 모델링 시스템 중 하나를 선택하여 모델링을 작성함

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 논리적 데이터 모델

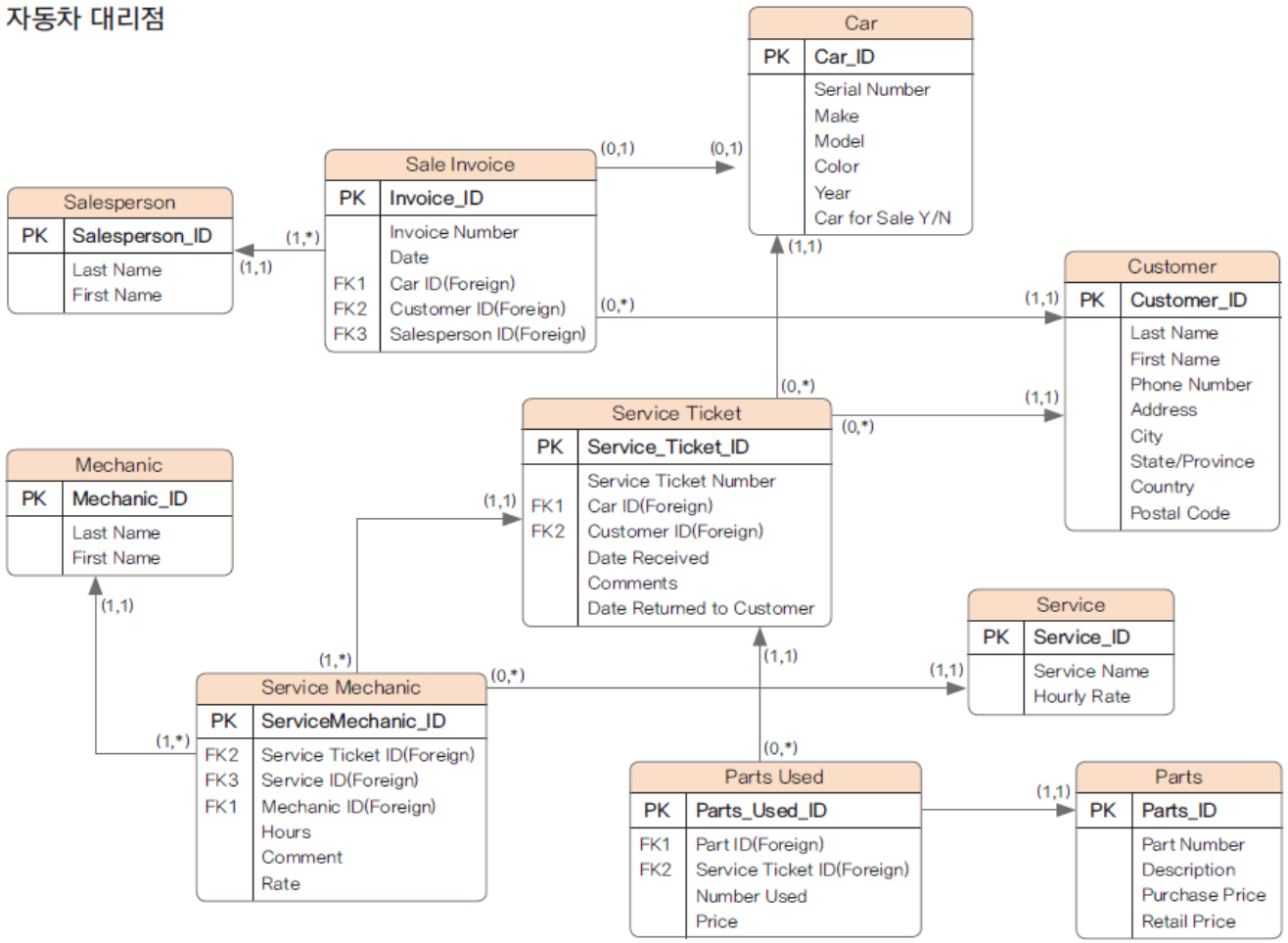
- 이렇게 생성된 결과물을 논리 모델, 다른 말로 논리적 ERD라고 함.
  논리적 ERD에는 다음 정보가 포함되어 있음

  - 다양한 속성의 데이터 유형(예: 문자형 또는 숫자형)
  - 데이터 엔티티 간의 관계
  - 데이터의 기본 속성 또는 기본키 정의

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 논리적 데이터 모델



자동차 대리점을 논리적 ERD로 표현한 예

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 논리적 데이터 모델

▪ 논리적 데이터 모델에서 도출할 수 있는 정보

- Showrooms 엔티티에는 문자열 데이터 유형인 '이름' 및 '위치' 필드와 숫자 데이터 유형인 '전화번호' 필드가 존재한다.
- Customers 엔티티에는 xxx@example.com 또는 xxx@example.com.yy 형식의 '이메일 주소' 필드가 있으며, 단, 이메일 주소는 100자 이하여야 한다.
- Sales 엔티티에는 '고객 이름' 필드, '판매원 이름' 필드, 날짜 데이터 유형의 '판매 날짜' 필드와 10진수 데이터 유형의 '금액' 필드가 존재한다.

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 논리적 데이터 모델

- 논리적 데이터 모델은 개발자가 개념적 데이터 모델을 바탕으로
  데이터베이스를 만드는데 사용할 기본적인 기술과 데이터베이스 언어 사이의 다리 역할을 함
- 이 모델은 DBMS의 종류에 구애받지 않으며 모든 데이터베이스 언어로 구현할 수 있음
- 데이터 엔지니어를 비롯한 관련자들은
  일반적으로 논리적 데이터 모델을 만든 후 어떤 데이터베이스 기술을 활용할 것인지 결정함

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 물리적 데이터 모델

- 논리적 데이터 모델을 관련자들이 **정한 DBMS 기술에 접목하고 해당 DB 언어를 사용하여** 만든다

- 논리적 데이터 모델링을 바탕으로 **실제 사용하게 될 DBMS 제조사(오라클, MS 등)를 선택하고 그에 맞는 물리적 모델링을 생성**함

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

▪ 물리적 데이터 모델

- 물리적 데이터 모델은
  논리적 데이터 모델과 최종적으로 선택할 DBMS 사이의
  다리 역할을 하며 다음과 같은 내용이 고려돼야 함

  - DBMS에 표현된 데이터 필드 유형이 제대로 정의되었는가?
  - 데이터 엔지니어가 최종 설계를 구현하기 전에 물리적 모델을 생성하였는가?
  - 공식 데이터 모델링 기술을 따라 설계의 모든 측면을 다뤘는지 확인하였는가?

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링의 유형

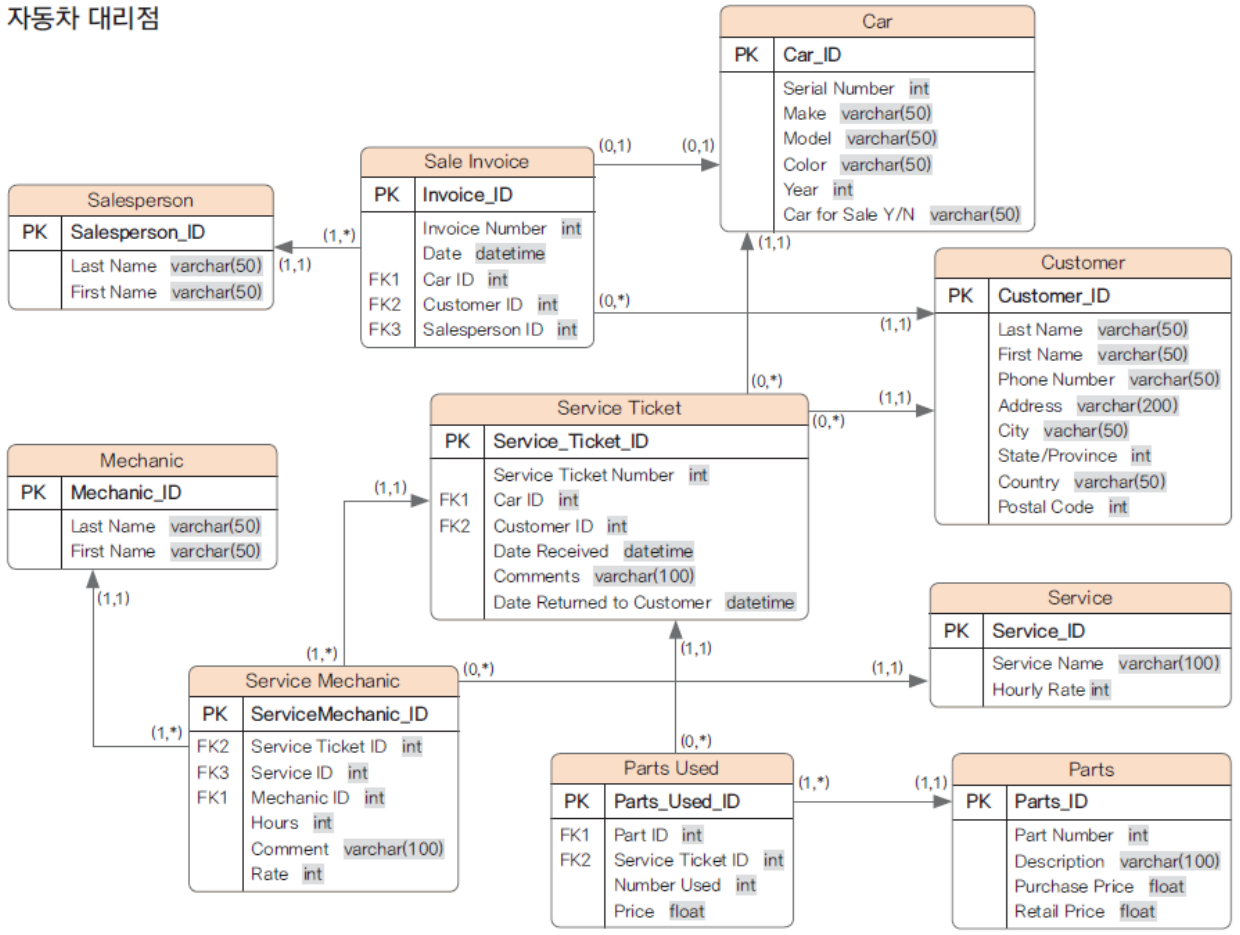
▪ 물리적 데이터 모델

- 자동차 대리점을 예로 들어
  물리적 데이터 모델에서 도출해야 하는 다양한 정보는 다음과 같음

  • Sales에서 '판매 금액'은 숫자 데이터 유형(float)이고 '판매 날짜'는 시간 데이터 유형(timestamp)으로 저장한다.
  • Customers에서 '고객 이름'은 문자열 데이터 유형으로 저장한다.

03-2 데이터베이스 모델링 이해하기

- **데이터 모델링의 유형**
  - **물리적 데이터 모델**



자동차 대리점을 물리적 ERD로 표현한 예

03-2 데이터베이스 모델링 이해하기

▪ 데이터 모델링 단계

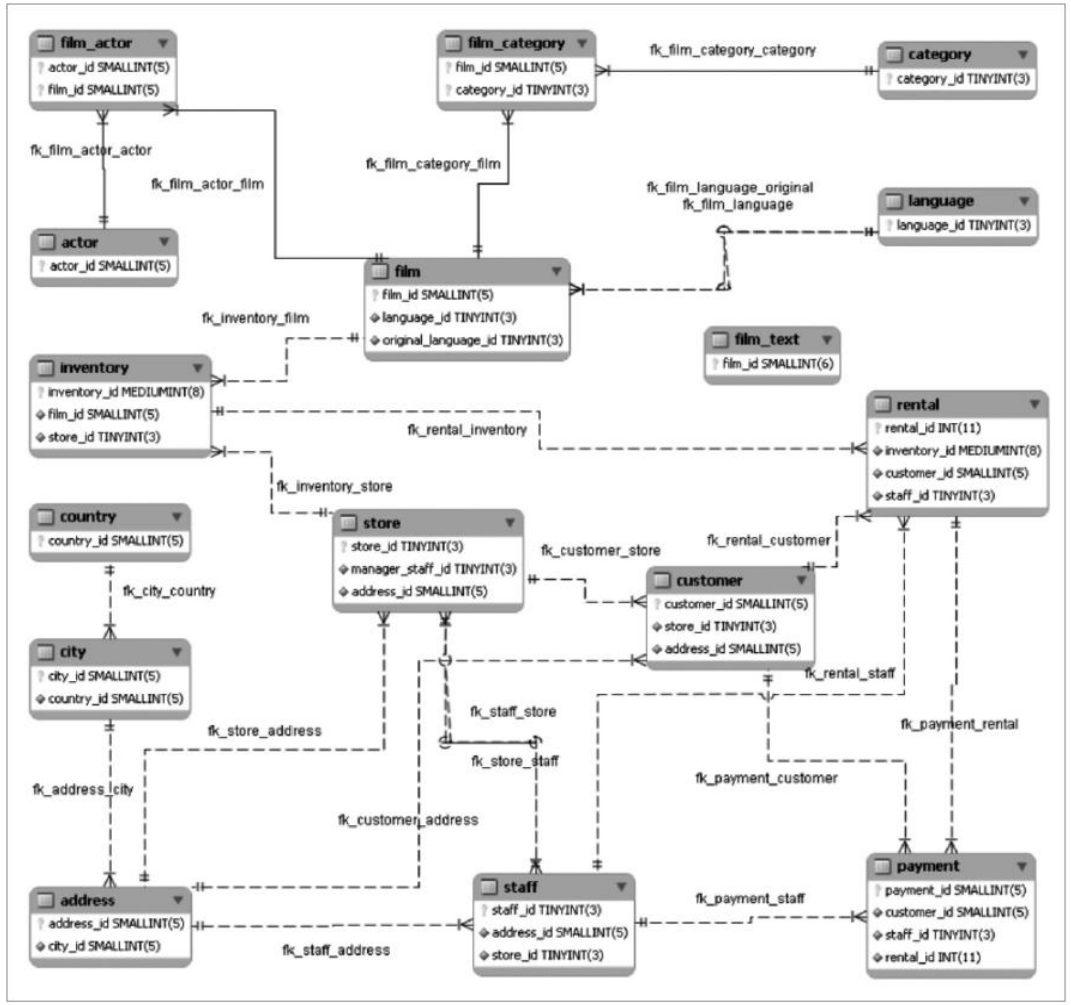
- 어떤 조직에서 데이터베이스를 구축할 때 데이터 모델링 과정
- 상위 개념으로(여기서는 오른쪽으로) 갈수록 구체화 · 세분화되고, 하위 개념으로(여기서는 왼쪽으로) 갈수록 추상화 · 단순화됨

03-2 데이터베이스 모델링 이해하기

- **sakila 데이터베이스 둘러보기**
  - 앞으로 실습에서 sakila라는 MySQL에서 제공하는 데이터베이스를 사용하려고 함
  - sakila 데이터베이스는 영화, 배우, 영화와 배우 간 관계, 상점, 대여를 연결하는 중앙 재고 테이블 등을 특징으로 하는 DVD 대여점을 주제로 한 데이터를 모델링한 것

03-2 데이터베이스 모델링 이해하기

- **sakila 데이터베이스 둘러보기**
  - 다음 그림은 sakila 데이터베이스의 ERD로, 릴레이션을 따라 테이블을 하나씩 살펴보면서 테이블의 데이터와 연관된 테이블 간의 데이터 관계 정도만 확인하자



sakila 데이터베이스의 ERD