

데이터베이스와 SQL

2장

데이터베이스 생성과 데이터 추가

빅데이터 분석가 과정

목차

- mysql 명령행 도구 사용 방법
- MySQL 자료형
- 테이블 생성
- 테이블 수정

mysql 명령행 도구 사용 방법

■ MySQL 명령행 사용 방법

- MySQL 8.0 Command Line Client 또는 DBeaver 실행
- 사용 가능한 데이터베이스 확인

```
mysql > show databases;
```

```
mysql> show databases;
```

Database
information_schema
mysql
performance_schema
sakila
sqlclass_db

```
+-----+
```

```
8 rows in set (0.01 sec)
```

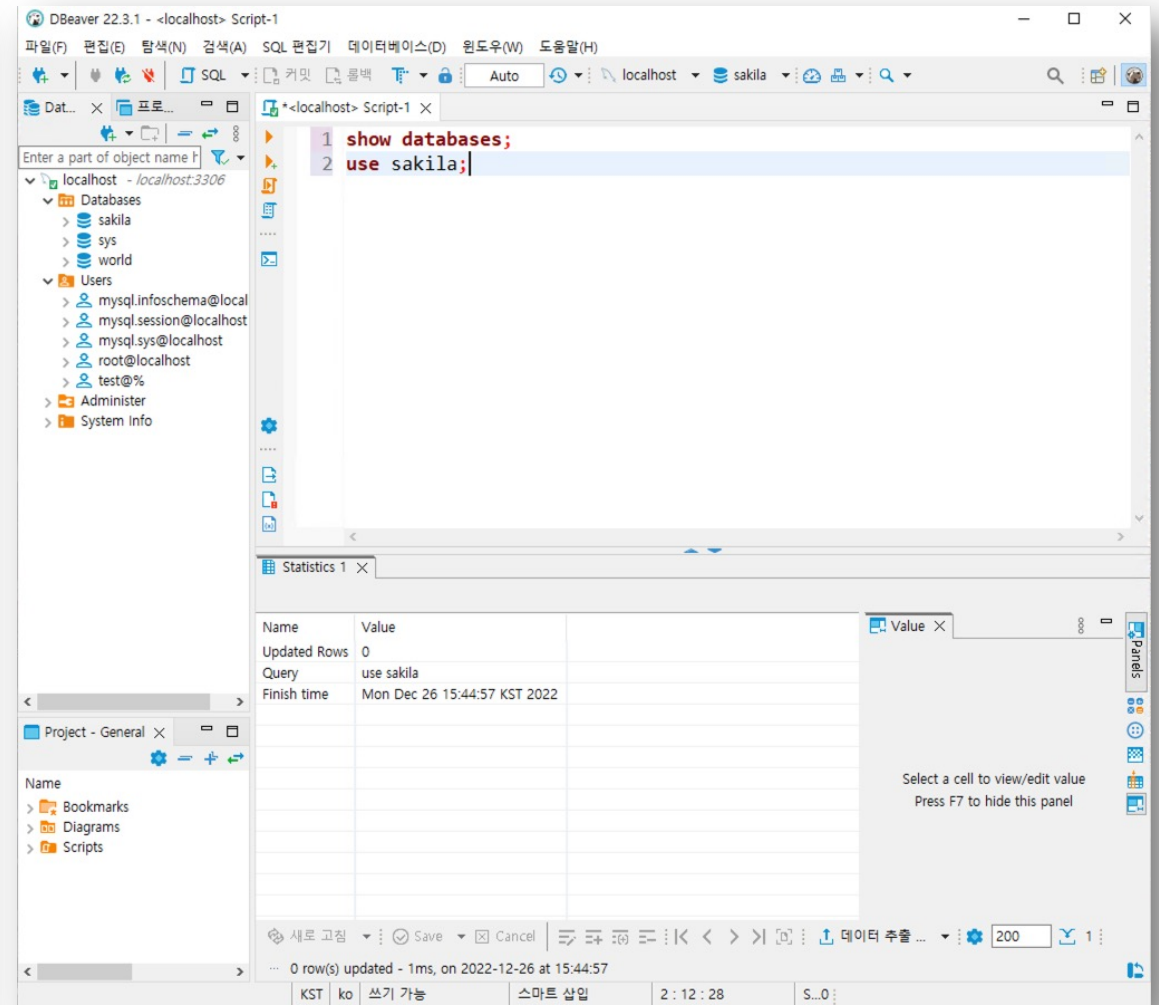
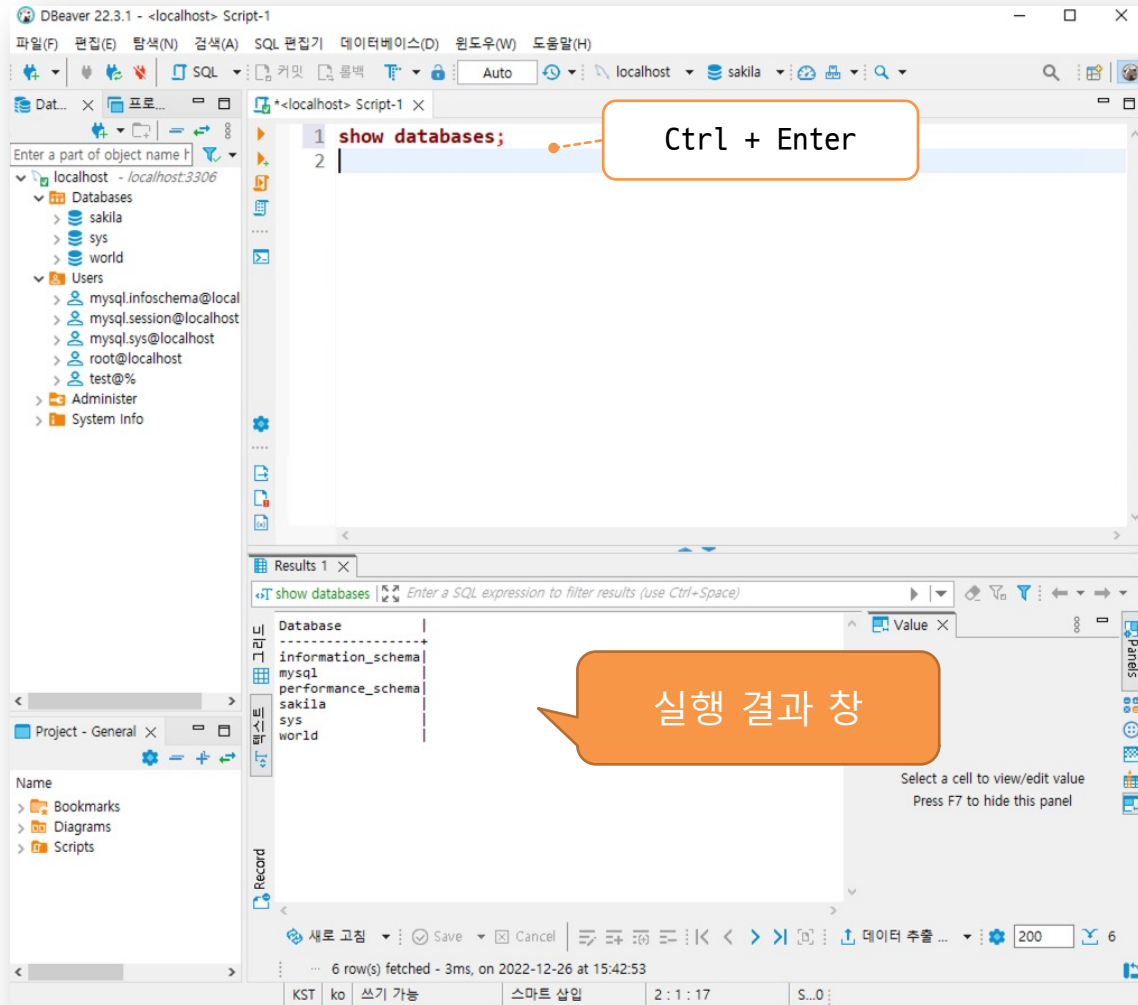
```
mysql>
```

- 샤키라(sakila) 데이터베이스 선택

```
mysql> use sakila;
```

DBeaver 사용 방법

- SQL 편집기 메뉴 > 새 SQL 편집기(Ctrl + J) 선택 후 직접 입력
- 명령행 한 줄 실행: 명령어 맨 뒤에서 Ctrl + Enter 키 입력



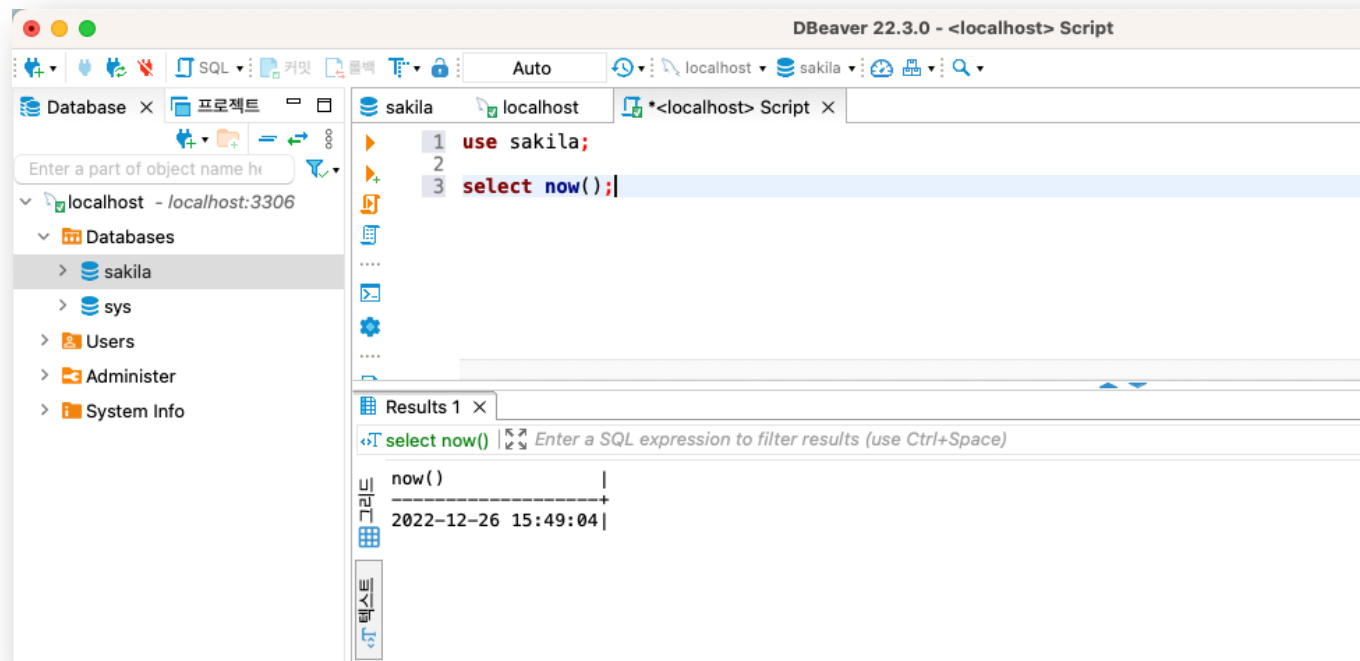
mysql 명령줄 도구 사용 방법

- 현재 날짜와 시간 정보 출력 쿼리

```
select now();
```

```
now()          |
-----+
2022-12-26 16:14:36|
```

- DBeaver에서 실행한 화면



MySQL 자료형

■ 문자 데이터

- 고정 길이 또는 가변 길이 문자열 저장

■ char(20)

- 고정 길이 문자열 (최대 255 바이트)
- 무조건 20바이트 차지 (20바이트가 넘으면 데이터는 잘림)
- 검색 속도 및 읽는 속도가 빠름

■ varchar(20)

- 가변 길이 문자열 (최대 65,535 바이트)
- 입력한 크기만큼의 공간만 차지 (20바이트는 넘으면 데이터 잘림)

MySQL 자료형

■ 텍스트 데이터

- varchar의 제한을 초과하는 데이터를 저장하는 자료형

자료형	최대 바이트 크기
tinytext	255 (1바이트)
text	65,535 (2바이트)
mediumtext	16,777,215 (24바이트)
longtext	4,294,967,295 (32바이트)

- 최대 크기를 초과하는 경우, 데이터가 잘려서 저장
- text 자료형과 varchar 자료형의 크기가 동일함
 - text: 최대 크기를 지정할 수 없음(무조건 65535 바이트의 공간을 차지)

MySQL 자료형

■ 숫자 데이터

■ 정수 자료형

자료형	부호 있는 정수 저장값의 범위	부호가 없는 정수 저장값의 범위
tinyint	-128 ~ 127	0 ~ 255
smallint	-32768 ~ 32,767	0 ~ 65,535
mediumint	-8,388,608 ~ 8,388,607	0 ~ 16,777,215
int	-2,147,483,648 ~ 2,147,483,647	0 ~ 4,294,967,295
bigint	$-2^{63} \sim 2^{63} - 1$	$0 \sim 2^{64} - 1$

■ 부동 소수점

자료형	숫자 범위
float(p, s)	$-3.402823466 \times 10^{38} \sim 3.402823466 \times 10^{38}$ ex) float(4, 2): 총 4자리 중 소수점 아래 2자리
double(p, s)	$-1.7976931348623157 \times 10^{308} \sim 1.7976931348623157 \times 10^{308}$

MySQL 자료형

■ 시간 데이터

자료형	기본 형식	허용값
date	YYYY-MM-DD	1000-01-01 ~ 9999-12-31 '2023-01-16' 형태의 문자열로 저장
datetime	YYYY-MM-DD HH:MI:SS	1000-01-01 00:00:00.000000 ~ 9999-12-31 23:59:59.999999
timestamp	YYYY-MM-DD HH:MI:SS	1970-01-01 00:00:00.000000 ~ 2038-01-18 22:14:07.999999
year	YYYY	1901 ~ 2155
time	HHH:MI:SS	-838:59:59.000000 ~ 838:59:59.000000

■ 날짜 형식의 구성 요소

자료형	기본 형식	허용값
YYYY	연도	1000 ~ 9999
MM	월	01(1월) ~ 12(12월)
DD	일	01 ~ 31
HH	시간	00 ~ 23
MI	분	00 ~ 59
SS	초	00 ~ 59

datetime과 timestamp 비교

■ datetime과 timestamp 비교 테이블

	datetime	timestamp
데이터 저장 타입	문자형	숫자형
저장 공간	8 bytes	4 bytes
자동 입력 여부	X	0
타임존 영향	X	0 (UTC로 변환)

- timestamp는 데이터를 입력하거나 수정하면 자동으로 시간 정보가 입력

새로운 데이터베이스 및 테이블 생성

- DBeaver로 MySQL접속

- 연습용 데이터베이스 생성: `testdb`

```
CREATE DATABASE testdb;
```

- 새롭게 생성한 데이터베이스인 `testdb` 선택(사용)

```
USE testdb;
```

테이블 생성

■ person 테이블 구성

열	자료형	허용값	비고
person_id	smallint (unsinged)		primary key (기본 키)
first_name	varchar(20)		
last_name	varchar(20)		
eye_color	char(2)	BR, BL, GR	
birth_date	date		
street	varchar(30)		
city	varchar(20)		
state	varchar(20)		
country	varchar(20)		
postal_code	varchar(20)		



person	
123	person_id
ABC	fname
ABC	lname
ABC	eye_color
ABC	birth_date
ABC	street
ABC	city
ABC	state
ABC	country
ABC	postal_code

■ favorite_food 테이블 구성

열	자료형	비고
person_id	smallint (unsinged)	foreign key (외래 키)
food	varchar(20)	

2.4 테이블 생성

■ person 테이블 생성

```
# person 테이블이 있으면 삭제
DROP TABLE IF EXISTS person;
CREATE TABLE person
  (person_id SMALLINT UNSIGNED,
   fname VARCHAR(20),
   lname VARCHAR(20),
   eye_color ENUM('BR', 'BL', 'GR'),
   birth_date DATE,
   street VARCHAR(30),
   city VARCHAR(20),
   state VARCHAR(20),
   country VARCHAR(20),
   postal_code VARCHAR(20),
   CONSTRAINT pk_person PRIMARY KEY (person_id)
  );
```

생략 가능

기본 키 제약 조건
(NOT NULL, UNIQUE)

■ CONSTRAINT [제약 조건 이름] PRIMARY KEY (필드이름)

- 기본 키(primary key)로 person_id 열을 선정
 - NOT NULL과 UNIQUE(중복 안됨) 제약 조건의 특징을 가짐
- 제약 조건(CONSTRAINT)
 - 데이터의 무결성을 지키기 위해, 데이터를 입력 받을 때 실행되는 검사 규칙
 - NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY

2.4 테이블 생성

■ person 테이블 확인

```
desc person;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint unsigned	NO	PRI		
fname	varchar(20)	YES			
lname	varchar(20)	YES			
eye_color	enum('BR','BL','GR')	YES			
birth_date	date	YES			
street	varchar(30)	YES			
city	varchar(20)	YES			
state	varchar(20)	YES			
country	varchar(20)	YES			
postal_code	varchar(20)	YES			

■ Null 항목

- 특정 열의 데이터 생략 여부
- NO: person_id 필드는 primary key이므로 반드시 값이 입력되어야 함
- Null값의 의미: 해당 사항 없음, 알 수 없음, 비어 있음

2.4 테이블 생성

favorite_food 테이블 생성

```
DROP TABLE IF EXISTS favorite_food;  
CREATE TABLE favorite_food  
  (person_id SMALLINT UNSIGNED,  
   food VARCHAR(20),  
   CONSTRAINT pk_favorite_food PRIMARY KEY (person_id, food),  
   CONSTRAINT fk_fav_food_person_id FOREIGN KEY (person_id) REFERENCES person(person_id)  
  );
```

생략 가능



- PRIMARY KEY(person_id, food)
 - 2개의 primary key 설정
- CONSTRAINT fk_fav_food_person_id FOREIGN KEY (person_id)
 - 외래 키(foreign key) 제약 조건
 - favorite_food 테이블에서 person_id의 값에 person 테이블에 있는 값만 포함되도록 제한
- REFERENCES 테이블이름 (필드이름)
 - 현재 테이블(favorite_food)에서 다른 테이블(person)의 필드(person_id)를 참조하는 것을 명시

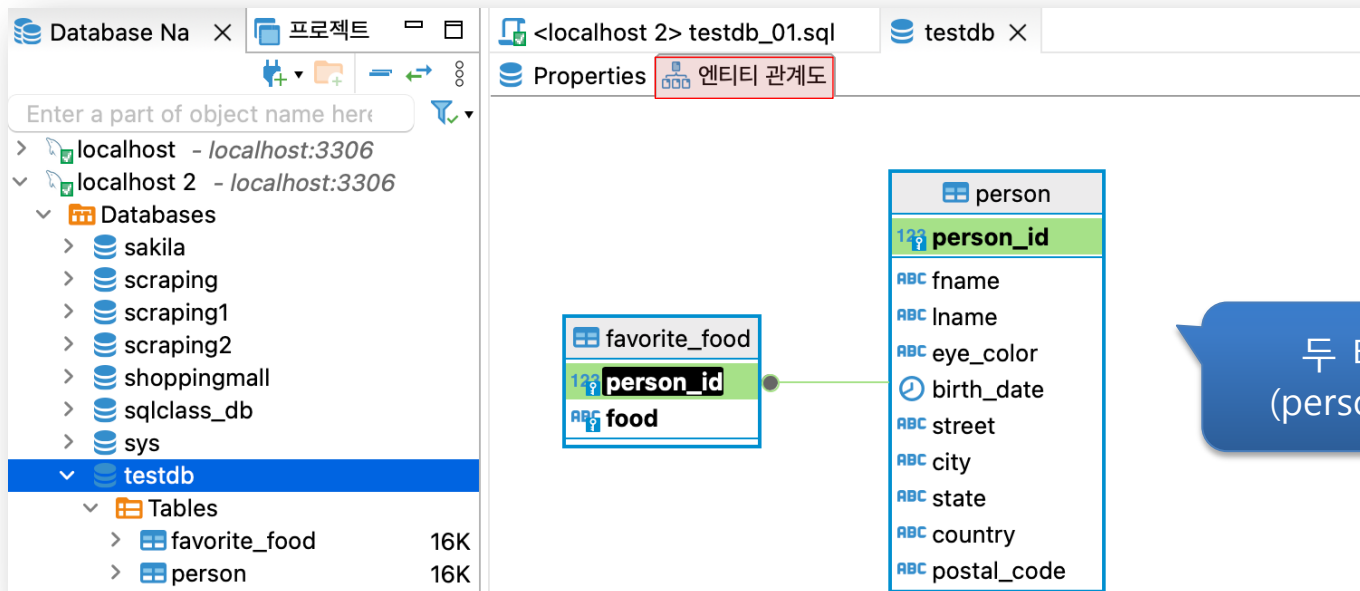
2.4 테이블 생성

favorite_food 테이블 확인

```
desc favorite_food;
```

Field	Type	Null	Key	Default	Extra
person_id	smallint unsigned	NO	PRI		
food	varchar(20)	NO	PRI		

DBeaver에서 엔티티 관계도 확인



2.5 테이블 수정 (ALTER)

■ 테이블 수정: ALTER

■ 숫자 키 데이터 생성

- MySQL: 자동 증가(auto-increment) 기능 제공
 - 데이터베이스 서버가 값을 제공

• SQL 명령어로 수정

ALTER TABLE 테이블명 MODIFY 컬럼명 데이터타입 추가할 내용;

• DBeaver의 Properties에서 변경

	컬럼명	#	Data Type	Not Null	Auto Increment
Columns	person_id	1	smallint unsigned	[v]	[v]
Constraints	fname	2	varchar(20)	[]	[]
Foreign Keys	lname	3	varchar(20)	[]	[]
References	eye_color	4	enum('BR','BL','G')	[]	[]
Triggers	birth_date	5	date	[]	[]
Indexes	street	6	varchar(30)	[]	[]
Partitions	city	7	varchar(20)	[]	[]
Statistics	state	8	varchar(20)	[]	[]
	country	9	varchar(20)	[]	[]
	postal_code	10	varchar(20)	[]	[]

person_id에 Null값을 주면,
자동으로 1부터 증가됨

2.5 테이블 수정 (ALTER)

- person 테이블의 person_id 재정의
 - person_id는 favorite_food에 외래키로 선정되어 있음, 제약 조건을 먼저 해제해야 됨
 - foreign key로 설정된 항목은 다른 테이블에서 변경 시 에러 발생
 - 제약 조건 비활성화 → 테이블 수정 → 제약 조건 활성화

```
set foreign_key_checks=0; # 제약 조건 비활성화
```

```
ALTER TABLE person MODIFY person_id smallint unsigned auto_increment;
```

```
set foreign_key_checks=1; # 제약 조건 활성화
```

Field	Type	Null	Key	Default	Extra
person_id	smallint unsigned	NO	PRI		auto_increment
fname	varchar(20)	YES			
lname	varchar(20)	YES			
eye_color	enum('BR', 'BL', 'GR')	YES			
birth_date	date	YES			
street	varchar(30)	YES			
city	varchar(20)	YES			
state	varchar(20)	YES			
country	varchar(20)	YES			
postal_code	varchar(20)	YES			

2.5 데이터 추가(INSERT)

■ 데이터 추가: INSERT 문

```
INSERT INTO 테이블이름 (열 이름1, 열 이름2, ...) VALUES (값1, 값2, ...);
```

```
INSERT INTO person
  (person_id, fname, lname, eye_color, birth_date)
VALUES (null, 'William', 'Turner', 'BR', '1972-05-27');
```

■ 데이터 확인: SELECT 문

```
SELECT * FROM 테이블이름;
```

■ 해당 테이블에서 모든 데이터(행, 컬럼) 데이터 출력: * 사용

```
SELECT * FROM person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27					

2.5 데이터 추가(INSERT)

- 데이터 확인: **SELECT** 문
 - 테이블의 특정 열의 데이터만 출력

```
SELECT 열 이름1, 열 이름2, ... FROM 테이블이름;
```

```
SELECT 열 이름1, 열 이름2, ... FROM 테이블이름 WHERE 열이름=값;
```

- person_id, fname, lname, birth_date 열 출력

```
select person_id, fname, lname, birth_date from person;
```

```
person_id|fname  |lname |birth_date|
-----+-----+-----+-----+
         1|William|Turner|1972-05-27|
```

- lname의 값이 'Turner'인 데이터에서 person_id, fname, lname, birth_date 열만 출력

```
select person_id, fname, lname, birth_date
from person where lname = 'Turner';
```

```
person_id|fname  |lname |birth_date|
-----+-----+-----+-----+
         1|William|Turner|1972-05-27|
```

2.5 데이터 추가 (INSERT)

- favorite_food 테이블에 데이터 추가
 - 한 행씩 추가

```
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'pizza');
```

```
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'cookies');
```

```
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'nachos');
```

- 한 번에 여러 행 추가
 - values (값1), (값2), ... ;

```
DELETE FROM favorite_food where person_id=1; # person_id값이 1인 데이터 삭제
```

```
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'pizza'),
       (1, 'cookie'),
       (1, 'nachos');
```

2.5 데이터 추가 (INSERT)

- favorite_food 테이블 데이터 확인
 - **ORDER BY** 컬럼이름: 컬럼의 값을 알파벳 순서로 정렬

```
SELECT food FROM favorite_food  
WHERE person_id=1 ORDER BY food;
```

```
food |  
-----+  
cookies|  
nachos |  
pizza  |
```

2.5 데이터 추가(INSERT)

- person 테이블에 다른 데이터 추가

```
INSERT INTO person
(person_id, fname, lname, eye_color, birth_date,
street, city, state, country, postal_code)
VALUES(null, 'Susan', 'Smith', 'BL', '1975-11-02',
'23 Maple St.', 'Arlington', 'VA', 'USA', '20220');
```

- person 테이블 데이터 확인

```
SELECT person_id, fname, lname, birth_date FROM person;
```

```
person_id|fname  |lname |birth_date|
-----+-----+-----+-----+
          1|William|Turner|1972-05-27|
          2|Susan  |Smith |1975-11-02|
```

- person_id 필드에 자동으로 2가 저장됨

2.5 데이터 수정 (UPDATE)

■ 데이터 수정: UPDATE 문

```
UPDATE 테이블이름 SET 필드이름1 = 값1, 필드이름2=값2, ...  
WHERE 필드이름=데이터값;
```

■ William Turner의 정보 추가

- William Turner의 자료 입력 과정에서 주소 정보는 입력하지 않았음
- UPDATE 문을 이용하여 주소 정보를 추가

```
UPDATE person  
SET street = '1225 Tremon St.',  
    city = 'Boston',  
    state = 'MA',  
    country = 'USA',  
    postal_code = '02138'  
WHERE person_id=1;
```

```
SELECT * FROM person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27	1225 Tremon St.	Boston	MA	USA	02138
2	Susan	Smith	BL	1975-11-02	23 Maple St.	Arlington	VA	USA	20220

2.5 데이터 삭제(DELETE)

■ 데이터 삭제: DELETE 문

```
DELETE FROM 테이블이름 WHERE 필드이름=데이터값;
```

- WHERE 절을 생략하면 테이블의 모든 데이터 삭제
 - 테이블은 삭제 되지 않음

```
DELETE FROM person WHERE person_id=2;
```

```
SELECT * FROM person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27	1225 Tremon St.	Boston	MA	USA	02138

■ 테이블 삭제: DROP TABLE 문

```
DROP TABLE 테이블이름;
```

- 다른 테이블에서 해당 테이블을 참조(foreign key)하는 상황에서는 에러 발생

2.6 오류 구문들

- 존재하지 않는 외래 키
 - 현재 person 테이블의 데이터

```
SELECT * FROM person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27	1225 Tremon St.	Bostyon	MA	USA	02138
2	Susan	Smith	BL	1975-11-02	23 Maple St.	Arlington	VA	USA	20220



person
123 person_id
ABC fname
ABC lname
ABC eye_color
ABC birth_date
ABC street
ABC city
ABC state
ABC country
ABC postal_code

- favorite_food 테이블에 데이터 추가

```
INSERT INTO favorite_food (person_id, food) VALUES (3, 'lasagna');
```

- 에러 발생
 - person_id=3의 값이 person 테이블에 존재하지 않음
 - 외래 키 제약 조건 위반
 - » favorite_food 테이블에 입력된 person_id의 모든 값이 person 테이블에 존재함을 보증
- person 테이블은 상위(parent)로 간주하고, favorite_food 테이블을 하위(child)로 간주
 - 상위 테이블인 person 테이블에 먼저 데이터를 입력 후, favorite_food 테이블에 추가해야 됨

2.6 오류 구문들

■ 외래 키 제약 조건 위반 해결

- person 테이블에 데이터 추가 (person_id=3 또는 person_id=null)

```
INSERT INTO person (person_id, fname, lname) VALUES(3, 'Kevin', 'Kern');
```

```
INSERT INTO person (person_id, fname, lname) VALUES(null, 'Kevin', 'Kern');
```

또는

```
SELECT * FROM person;
```

person_id	fname	lname	eye_color	birth_date	street	city	state	country	postal_code
1	William	Turner	BR	1972-05-27	1225 Tremon St.	Bostyon	MA	USA	02138
2	Susan	Smith	BL	1975-11-02	23 Maple St.	Arlington	VA	USA	20220
3	Kevin	Kern							

- favorite_food 테이블에 데이터 추가

```
INSERT INTO favorite_food (person_id, food) VALUES (3, 'lasagna');
```

```
SELECT * FROM favorite_food;
```

person_id	food
1	cookie
1	nachos
1	pizza
3	lasagna

person 테이블에 person_id값 3이 존재함

2.6 오류 구문들

- 잘못된 날짜 변환

- 날짜 기본 형식: YYYY-MM-DD

```
UPDATE person SET birth_date = 'DEC-21-1980' WHERE person_id=1;
```

- 오류 발생: Incorrect data value

- str_to_date(str, format) 함수: 7장

- 문자열을 format(형식 문자열)을 사용하여 DATETIME 값으로 변환 후 반환

```
UPDATE person SET birth_date = str_to_date('DEC-21-1980', '%b-%d-%Y')  
WHERE person_id=1;
```

- 'DEC' -> '%b': short month name(Jan, Feb, ...)

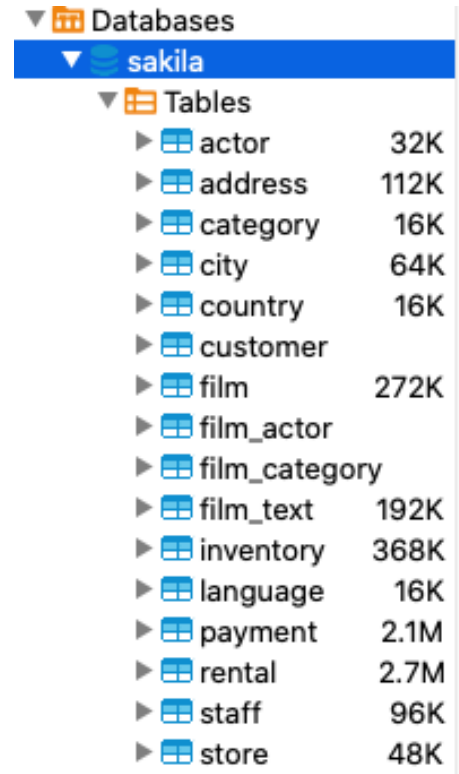
- '21' -> '%d': day(00 .. 31)

- '1980' -> '%Y': 4자리 연도

2.7 샤키라 데이터베이스

- 샤키라 데이터베이스
 - MySQL에서 샘플로 제공하는 데이터베이스
 - DVD 대여점 체인을 설계
 - sakila 테이블 이름 및 정의

테이블명	정의
film	출시되어 대여할 수 있는 영화
actor	영화에 출연하는 배우
customer	영화를 보는 고객
category	영화 장르
payment	고객이 지불한 영화 대여료
language	영화배우들이 말하는 언어
film_actor	영화속 배우
inventory	대여 가능한 영화 여부



▼ Databases
▼ sakila
▼ Tables
▶ actor 32K
▶ address 112K
▶ category 16K
▶ city 64K
▶ country 16K
▶ customer
▶ film 272K
▶ film_actor
▶ film_category
▶ film_text 192K
▶ inventory 368K
▶ language 16K
▶ payment 2.1M
▶ rental 2.7M
▶ staff 96K
▶ store 48K

2.7 샤키라 데이터베이스

- sakila 데이터베이스 사용 및 sakila 데이터베이스에 포함된 테이블 확인

```
USE sakila;  
SHOW tables;
```

- customer 테이블 구성 확인

```
desc customer;
```

Field	Type	Null	Key	Default	Extra
customer_id	smallint unsigned	NO	PRI		auto_increment
store_id	tinyint unsigned	NO	MUL		
first_name	varchar(45)	NO			
last_name	varchar(45)	NO	MUL		
email	varchar(50)	YES			
address_id	smallint unsigned	NO	MUL		
active	tinyint(1)	NO		1	
create_date	datetime	NO			
last_update	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT_TIMESTAMP

	ABC Tables_in_sakila
1	actor
2	actor_info
3	address
4	category
5	city
6	country
7	customer
8	customer_list
9	film
10	film_actor
11	film_category
12	film_list
13	film_text
14	inventory
15	language
16	nicer_but_slower_film_list
17	payment
18	rental
19	sales_by_film_category
20	sales_by_store
21	staff
22	staff_list
23	store



Questions?