0. AI - SQL 추출

#0.강의/2.데이터베이스로드맵/0.준비

다음 [강의 내용]에서 지시한 내용을 추출해라.

1. 강의 내용은 마크다운 형식이다.
2. `sql로 시작하고` 으로 끝나는 내용은 모두 추출해라. `sql,` 은 출력결과에 그대로 포함해라.
3. #, ##으로 시작하는 제목은 반드시 모두 추출해라. 제목 추출시 앞에 -- 붙이고 #은 유지해라. 추출후 마지막 부분에 반드시 엔터를 입력해라.
4. ###, ####, #####으로 시작하는 제목은 `sql로 시작하고,` 으로 끝나는 내용이 있는 경우에만 출력 결과에 포함해라. 그렇지 않으면 무시해라.
5. "시작하는 내용은 무시해라.

[입력 예시]

2. 공통 코드 설계

#0.강의/2.데이터베이스로드맵/4.설계2

- /공통 코드가 필요한 이유
- /공통 코드 테이블 설계
- /공통 코드를 더 범용성 있게 - 그룹화 설계
- /공통 코드와 추가 속성
- /공통 코드의 단점
- /공통 코드의 단점 해결 방안 1
- /공통 코드의 단점 해결 방안 2
- /공통 코드 vs 애플리케이션 ENUM 1
- /공통 코드 vs 애플리케이션 ENUM 2
- /공통 코드 vs 애플리케이션 ENUM 3
- /공통 코드 설계와 비즈니스 설계의 차이
- /정리

공통 코드가 필요한 이유

데이터베이스를 설계하다 보면 비슷한 종류의 데이터가 여러 테이블에 걸쳐 반복되는 상황을 자주 만난다. 예를 들어 주

문 상태, 회원 등급, 결제 방법 같은 것들이다. 이런 데이터를 어떻게 관리해야 할까? 먼저 문제 상황부터 살펴보자.

실습 준비

```
-- 데이터베이스가 존재하지 않으면 생성
CREATE DATABASE IF NOT EXISTS my_shop4;
USE my_shop4;
```

- 실습에서는 `my_shop4` 데이터베이스를 사용하겠다.

문제 상황 - 하드코딩된 상태값

쇼핑몰을 운영한다고 가정하자. 주문 테이블에는 주문 상태가 필요하다. 가장 단순한 방법은 상태값을 문자열로 직접 저장하는 것이다.

```
DROP TABLE IF EXISTS orders;

CREATE TABLE orders (
    order_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    member_id BIGINT NOT NULL,
    order_status VARCHAR(20) NOT NULL,
    total_amount INT NOT NULL,
    created_at DATETIME NOT NULL
);
```

주문 데이터를 입력해보자.

```
INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(1, '주문완료', 50000, '2026-01-15 10:30:00'),
(2, '결제완료', 75000, '2026-01-15 11:00:00'),
(3, '배송중', 30000, '2026-01-15 12:00:00'),
(1, '배송완료', 120000, '2026-01-14 09:00:00'),
(4, '주문취소', 45000, '2026-01-13 15:00:00');
```

```
SELECT * FROM orders;
```

[실행 결과]

order_id	member_id	order_status	total_amount	created_at
1	1	주문완료	50000	2026-01-15 10:30:00
2	2	결제완료	75000	2026-01-15 11:00:00
3	3	배송중	30000	2026-01-15 12:00:00
4	1	배송완료	120000	2026-01-14 09:00:00
5	4	주문취소	45000	2026-01-13 15:00:00

언뜻 보기에는 문제가 없어 보인다. 하지만 실무에서 이 방식을 그대로 사용하면 여러 가지 심각한 문제가 발생한다.

문제1: 데이터 불일치

개발자 A는 주문 상태를 '주문완료'로 입력하고, 개발자 B는 '주문 완료'(띄어쓰기 포함)로 입력한다. 또 다른 개발자 C는 'ORDER_COMPLETE'로 입력한다.

```
INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(5, '주문 완료', 80000, '2026-01-16 10:00:00'),
(6, 'ORDER_COMPLETE', 65000, '2026-01-16 11:00:00');
```

```
SELECT order_id, order_status FROM orders;
```

[실행 결과]

order_id	order_status
1	주문완료
2	결제완료
3	배송중

4	배송완료
5	주문취소
6	주문 완료
7	ORDER_COMPLETE

같은 의미인데 서로 다른 값이 저장되었다. 이제 '주문완료' 상태인 주문을 조회하면 어떻게 될까?

```
SELECT * FROM orders WHERE order_status = '주문완료';
```

[실행 결과]

order_id	member_id	order_status	total_amount	created_at
1	1	주문완료	50000	2026-01-15 10:30:00

분명히 3건이 주문완료 상태인데, 1건만 조회된다. 데이터 불일치로 인해 비즈니스 로직에 오류가 발생한 것이다.

문제2: 오타로 인한 버그

상태값을 직접 문자열로 입력하면 오타가 발생할 수 있다.

```
INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(7, '배송증', 55000, '2026-01-17 09:00:00');  -- '배송중'을 '배송증'으로 오타
```

이 데이터는 정상적으로 저장된다. 데이터베이스는 '배송증'이 오타인지 알 수 없기 때문이다. 이런 오타는 나중에 발견하기 매우 어렵고, 발견하더라도 이미 많은 데이터가 잘못 저장된 후일 수 있다.

문제3: 상태 변경의 어려움

기획팀에서 '주문완료'라는 단어가 명확하지 않다고 앞으로는 '주문완료'라는 단어 대신에 '주문성공'이라는 단어로 상태를 변경하자고 요청한다. 이 경우 이미 저장된 모든 데이터를 UPDATE 해야 한다.

```sql
SET SQL_SAFE_UPDATES = 0; -- 안전 업데이트 모드 끄기

UPDATE orders SET order_status = '주문성공' WHERE order_status = '주문완료';

SET SQL_SAFE_UPDATES = 1; -- 안전 업데이트 모드 활성화
```

> **MySQL 워크벤치 안전 업데이트 모드**
> MySQL 워크벤치는 실수로 인한 대량의 데이터 변경을 예방하기 위해 안전모드를 제공한다.
> WHERE 문에 키를 제공하거나 또는 안전 업데이트 모드를 꺼야 UPDATE, DELETE 문을 실행할 수 있다. (입문편 참고)

데이터가 수백만 건이라면? 이 UPDATE 문은 매우 오래 걸리고, 그 동안 테이블에 락이 걸려 서비스에 영향을 줄 수 있다. 그리고 앞서 본 것처럼 '주문 완료(띄어쓰기 포함)', 'ORDER_COMPLETE' 같은 변형된 데이터는 UPDATE 되지 않는다.

문제4: 표시 이름과 코드값의 혼재
화면에 표시되는 이름을 데이터베이스에 직접 저장하면 또 다른 문제가 생긴다. 만약 영어 서비스를 추가해야 한다면? '주문완료'를 'Order Complete'로 표시해야 하는데, 데이터베이스에 한글로 저장되어 있으면 처리가 복잡해진다.

해결책 - 코드값 분리

이 문제들을 해결하려면 **코드값**과 **표시 이름**을 분리해야 한다. 데이터베이스에는 코드값만 저장하고, 화면에 표시할 이름은 별도로 관리하는 것이다.

먼저 주문 테이블을 다시 만들어보자.

```sql
DROP TABLE IF EXISTS orders;

CREATE TABLE orders (
    order_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    member_id BIGINT NOT NULL,
    order_status VARCHAR(20) NOT NULL,
    total_amount INT NOT NULL,
    created_at DATETIME NOT NULL
);
```

이번에는 상태값을 영문 코드로 저장한다.

```
INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(1, 'ORDER', 50000, '2026-01-15 10:30:00'),
(2, 'PAID', 75000, '2026-01-15 11:00:00'),
(3, 'SHIPPING', 30000, '2026-01-15 12:00:00'),
(1, 'DELIVERED', 120000, '2026-01-14 09:00:00'),
(4, 'CANCEL', 45000, '2026-01-13 15:00:00');
```

```
SELECT * FROM orders;
```

[실행 결과]

order_id	member_id	order_status	total_amount	created_at
1	1	ORDER	50000	2026-01-15 10:30:00
2	2	PAID	75000	2026-01-15 11:00:00
3	3	SHIPPING	30000	2026-01-15 12:00:00
4	1	DELIVERED	120000	2026-01-14 09:00:00
5	4	CANCEL	45000	2026-01-13 15:00:00

그리고 코드값과 표시 이름의 매핑 정보는 별도로 관리한다. 가장 단순한 형태는 다음과 같다.

코드값	한글 이름	영문 이름
ORDER	주문시작	Order Received
PAID	결제완료	Payment Complete
SHIPPING	배송중	Shipping

DELIVERED	배송완료	Delivered
CANCEL	주문취소	Cancelled

이 매핑 정보를 어디에 저장할까? 바로 **공통 코드 테이블**이다.

[출력 예시]

```
-- # 2. 공통 코드 설계
-- #0.강의/2.데이터베이스로드맵/4.설계2

-- ## 공통 코드가 필요한 이유

-- ### 실습 준비

-- 데이터베이스가 존재하지 않으면 생성
CREATE DATABASE IF NOT EXISTS my_shop4;
USE my_shop4;

-- ### 문제 상황 - 하드코딩된 상태값
DROP TABLE IF EXISTS orders;

CREATE TABLE orders (
    order_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    member_id BIGINT NOT NULL,
    order_status VARCHAR(20) NOT NULL,
    total_amount INT NOT NULL,
    created_at DATETIME NOT NULL
);


INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(1, '주문완료', 50000, '2026-01-15 10:30:00'),
(2, '결제완료', 75000, '2026-01-15 11:00:00'),
(3, '배송중', 30000, '2026-01-15 12:00:00'),
```

```
(1, '배송완료', 120000, '2026-01-14 09:00:00'),
(4, '주문취소', 45000, '2026-01-13 15:00:00');

SELECT * FROM orders;


-- #### 문제1: 데이터 불일치

INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(5, '주문 완료', 80000, '2026-01-16 10:00:00'),
(6, 'ORDER_COMPLETE', 65000, '2026-01-16 11:00:00');

SELECT order_id, order_status FROM orders;

SELECT * FROM orders WHERE order_status = '주문완료';


-- #### 문제2: 오타로 인한 버그

INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(7, '배송증', 55000, '2026-01-17 09:00:00');  -- '배송중'을 '배송증'으로 오타

-- #### 문제3: 상태 변경의 어려움

SET SQL_SAFE_UPDATES = 0; -- 안전 업데이트 모드 끄기

UPDATE orders SET order_status = '주문성공' WHERE order_status = '주문완료';

SET SQL_SAFE_UPDATES = 1; -- 안전 업데이트 모드 활성화

-- #### 문제4: 표시 이름과 코드값의 혼재

-- ### 해결책 - 코드값 분리

DROP TABLE IF EXISTS orders;

CREATE TABLE orders (
```

```
    order_id BIGINT PRIMARY KEY AUTO_INCREMENT,
    member_id BIGINT NOT NULL,
    order_status VARCHAR(20) NOT NULL,
    total_amount INT NOT NULL,
    created_at DATETIME NOT NULL
);

INSERT INTO orders (member_id, order_status, total_amount, created_at) VALUES
(1, 'ORDER', 50000, '2026-01-15 10:30:00'),
(2, 'PAID', 75000, '2026-01-15 11:00:00'),
(3, 'SHIPPING', 30000, '2026-01-15 12:00:00'),
(1, 'DELIVERED', 120000, '2026-01-14 09:00:00'),
(4, 'CANCEL', 45000, '2026-01-13 15:00:00');

SELECT * FROM orders;
```

[강의 내용]