어느 쇼핑몰 사이트의 데이터베이스 안에는 고객들의 주문 정보를 담고 있는 **customer\_order 테이블**과, 각 주문의 현재 배송 상태 정보를 담고 있는 **delivery 테이블**이 있다고 합시다.

각 테이블은 다음과 같이 생겼는데요.

**customer\_order 테이블**

| **id** | **item\_no** | **payment\_id** | **approval\_time** |
| --- | --- | --- | --- |
| ... | ... | ... | ... |
| 10000 | 102 | 1304 | 2020-06-02 17:42:29 |
| 10001 | 304 | 1305 | 2020-06-02 17;50:04 |
| 10002 | 2091 | 1306 | 2020-06-02 18:00:51 |
| ... | ... | ... | ... |

**delivery 테이블**

| **id** | **order\_id** | **location\_code** | **status\_code** | **recent\_update\_time** |
| --- | --- | --- | --- | --- |
| ... | ... | ... | ... | ... |
| 9983 | 10000 | 1 | 0 | 2020-06-03 18:00:23 |
| 9984 | 10001 | 3 | 1 | 2020-06-04 15:23:27 |
| 9985 | 10002 | 4 | 2 | 2020-06-05 09:12:03 |
| ... | ... | ... | ... | ... |

지금 delivery 테이블의 order\_id 컬럼을, customer\_order 테이블의 id 컬럼을 참조하는 물리적 **Foreign Key**로 설정하려고 합니다.

SQL 문을 써서 직접 Foreign Key를 설정해보세요. 이때 Foreign Key의 이름은 **fk\_delivery\_order**로 지으시고,

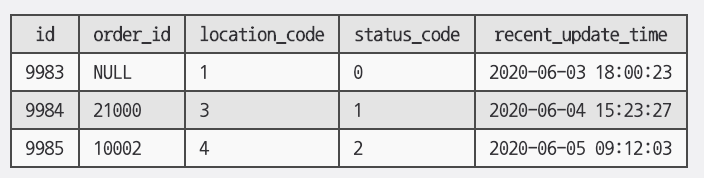
부모 테이블의 row가 삭제되거나, Foreign Key에 의해 참조당하는 컬럼의 값이 갱신될 때에 관한 정책은

**ON DELETE**의 경우 그 row를 참조하던 자식 테이블의 row들의 컬럼에 NULL이 들어가도록 설정하고,

**ON UPDATE**의 경우 그 row를 참조하던 자식 테이블의 row들의 컬럼도 같은 값으로 갱신되도록 설정하세요.

오른쪽 템플릿에는 여러분이 정책을 잘 설정했는지 테스트하는 SQL 코드들이 써있습니다. 여러분이 정답을 입력하고 실행하면 아래 출력 예시와 같은 결과를 보게 되실 겁니다.

**출력 예시**

**﻿**

ALTER TABLE delivery

ADD CONSTRAINT `fk\_delivery\_order`

FOREIGN KEY (order\_id)

REFERENCES customer\_order (id)

ON DELETE SET NULL

ON UPDATE CASCADE;

DELETE FROM customer\_order WHERE id = 10000;

UPDATE customer\_order SET id = 21000 WHERE id = 10001;

SELECT \* FROM delivery;