05

조인으로 두 테이블 묶기

05-1 조인의 종류

- 지금까지는 한 테이블에서만 데이터를 조회하는 쿼리를 작성해 봄
- 이번에는 테이블 2개에서 데이터를 조회하는
  쿼리를 작성하는 방법들 가운데
  가장 유용하게 활용할 수 있는 조인(JOIN)에 대해 알아보자
- 종류 - 내부조인, 외부조인, 교차조인, 셀프조인

05-1 조인의 종류

▪ 조인의 의미

- sakila 데이터베이스의
  customer 테이블에는 이름, 이메일 등 고객 정보가 저장되어 있고,
  address 테이블에는 거리 이름, 우편번호, 전화번호 등의 정보가 저장되어 있음

customer	
customer_id	고객 번호
store_id	스토어 번호
first_name	이름
last_name	성
email	이메일
address_id	주소 번호
active	활성화 유무
create_date	데이터 입력 시간
last_update	데이터 수정 시간

address	
address_id	주소 번호
address	상세 주소1
address2	상세 주소2
district	거리 이름
city_id	도시 번호
postal_code	우편번호
phone	전화번호
location	지역
last_update	데이터 수정 시간

05-1 조인의 종류

▪ 조인의 의미

- 고객 정보와 함께 주소를 조회하고 싶다면
  다음 표처럼 각 테이블의 데이터를 조합해보자

customer_id	store_id	(…생략…)	address	postal_code
1	1	…	1913 Hanoi Way	35200
2	1	…	1121 Loja Avenue	17886
3	1	…	692 Joliet Street	83579

- 이럴 때 사용하는 것이 바로 조인임
- 조인은 테이블 A의 열을 테이블 B에 포함하여 조회할 수 있게 만들어 줌

05-1 조인의 종류

▪ 조인의 의미

- 그런데 테이블을 만들 때 처음부터 customer 테이블에 address, postal_code 열을 만들면 조인할 필요가 없음
- 하지만 그렇게 테이블을 만들면 customer 테이블에도 address 열이 있고 address 테이블에도 address 열이 있는 셈이니 데이터가 중복으로 저장되는 문제가 생김

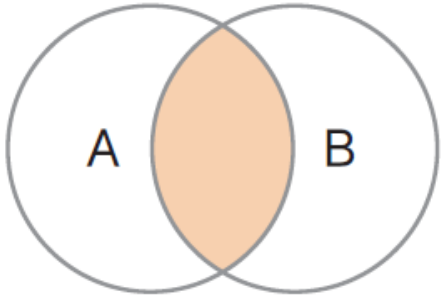
- 데이터가 중복으로 저장되면
  데이터를 저장할 공간이 더 필요하게 되고,
  중복 저장한 열 데이터를 수정할 때 해당 열이 있는 테이블의 데이터를
  모두 찾아 수정해야 하는 문제가 생김

- 데이터를 모델링할 때
  데이터의 중복을 최소화하는 것이 중요하므로
  조인을 사용하여 테이블 2개 이상을 조합한 결과를 조회하는 것이 효율적인 방식임

05-1 조인의 종류

▪ 내부 조인

- 보통 조인이라고 하면 내부 조인(INNER JOIN)을 의미할 정도로 가장 많이 씀
- 내부 조인으로는 두 테이블 모두 조건에 맞은 열을 조회할 수 있음
- 다이어그램으로 표현하면 다음과 같음



내부 조인 벤 다이어그램

05-1 조인의 종류

▪ 내부 조인

- 예. 고객 테이블에는 고객 번호, 고객 이름이 저장되어 있고,
  주문 테이블에는 주문 번호, 고객 번호, 주문 날짜가 저장되어 있음
- 만약 고객의 주문 내역을 확인하고 싶다면 어떻게 해야 할까?
- 이 두 테이블을 조인하려면 두 개의 테이블에 공통 열인 '고객 번호'를 조인키로 사용하면 해결이 가능함

고객 테이블

고객 번호	고객 이름
0001	한성우
0002	신지은
0003	이시연

주문 테이블

주문 번호	고객 번호	주문 날짜
100	0001	2024-02-16
101	0002	2024-02-17
102	0003	2024-02-18

내부 조인

고객 번호	고객 이름	주문 번호	고객 번호	주문 날짜
0001	한성우	100	0001	2024-02-16
0002	신지은	101	0002	2024-02-17

05-1 조인의 종류

▪ 내부 조인

INNER JOIN 문의 기본 형식

```
SELECT
        [열]
FROM [테이블 1]
        INNER JOIN [테이블 2] ON [테이블 1.열] = [테이블 2.열]
WHERE [검색 조건]
```

Do it! INNER JOIN 문을 적용한 예

```
SELECT
        [고객.고객 번호], [고객.고객 이름], [주문.주문 번호], [주문.고객 번호], [주문.주문 날짜]
FROM [고객]
        INNER JOIN [주문] ON [고객.고객 번호] = [주문.고객 번호]
```

05-1 조인의 종류

▪ 내부 조인

▪ INNER JOIN 문으로 테이블 2개 조인하기

customer 테이블과 address 테이블을 INNER JOIN 문으로 조인하여 first_name이 ROSA인 데이터를 조회하는 쿼리를 만들자

customer	
customer_id	고객 번호
store_id	스토어 번호
first_name	이름
last_name	성
email	이메일
address_id	주소 번호
active	활성화 유무
create_date	데이터 입력 시간
last_update	데이터 수정 시간

조인키

address	
address_id	주소 번호
address	상세 주소1
address2	상세 주소2
district	거리 이름
city_id	도시 번호
postal_code	우편번호
phone	전화번호
location	지역
last_update	데이터 수정 시간

05-1 조인의 종류

▪ 내부 조인

▪ INNER JOIN 문으로 테이블 2개 조인하기

Do it! 내부 조인한 테이블에서 조건에 맞는 데이터 조회

```
SELECT
        a.customer_id, a.store_id, a.first_name, a.last_name, a.email, a.address_id AS a_address_id,
        b.address_id AS b_address_id, b.address, b.district, b.city_id, b.postal_code, b.phone, b.location
FROM customer AS a
        INNER JOIN address AS b ON a.address_id = b.address_id
WHERE a.first_name = 'ROSA';
```

실행 결과

	customer_id	store_id	first_name	last_name	email	a_address_id	b_address_id	address	district	city_id	postal_code	phone
▶	112	2	ROSA	REYNOLDS	ROSA.REYNOLDS@sakilacustomer.org	116	116	793 Cam Ranh Avenue	California	292	87057	824370924746

05-1 조인의 종류

- **내부 조인**
  - **INNER JOIN 문으로 테이블 2개 조인하기**
    - 테이블의 열 이름이 유일하다면 별칭을 사용하지 않아도 되지만
      현재는 조인할 두 테이블에 이름이 같은 열이 있으므로
      이와 같이 a.address_id, b.address_id라고
      각 테이블의 별칭(AS)을 붙여서 조회하도록 하였고,
    - ON 문은 테이블을 조인할 때 조인키라고 하는 조인 조건으로 사용할 열을 지정함
      여기서는 각 테이블의 address_id 열을 조인 조건(조인키)으로 사용함

05-1 조인의 종류

- **내부 조인**
  - **INNER JOIN 문에 조인 조건 2개 이상 사용하기**
    - 조인 조건으로 열을 2개 이상 사용할 수도 있음.
    - AND, OR 등의 연산자를 사용하여 여러 조건을 조합할 수도 있음

05-1 조인의 종류

▪ 내부 조인

▪ INNER JOIN 문에 조인 조건 2개 이상 사용하기

customer와 address 각 테이블의 address_id와 customer 테이블의 create_date 열, address 테이블의 last_ last_update 열을 사용해 INNER JOIN하여 조회하는 쿼리

Do it! 2개의 조인 조건으로 조인한 테이블에서 조건에 맞는 데이터 조회

```
SELECT
        a.customer_id, a.first_name, a.last_name,
        b.address_id, b.address, b.district, b.postal_code
FROM customer AS a
        INNER join address AS b ON a.address_id = b.address_id AND a.create_date = b.last_update
WHERE a.first_name = 'ROSA';
```

실행 결과

	customer_id	first_name	last_name	address_id	address	district	postal_code

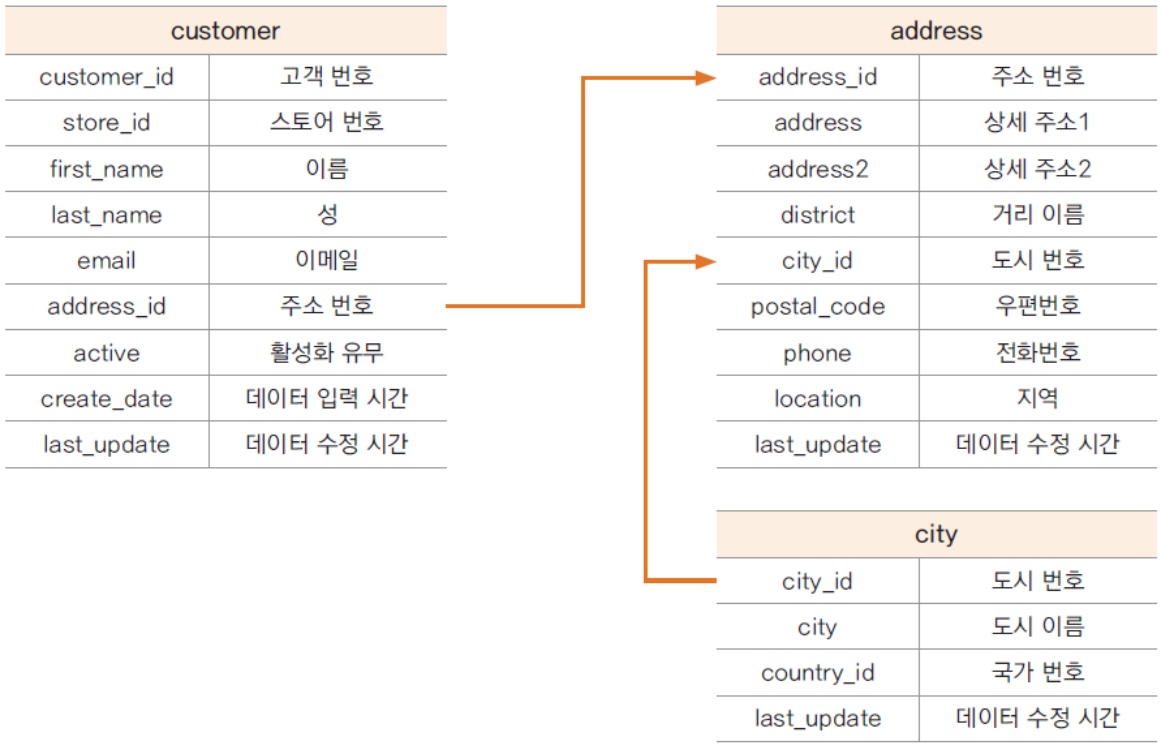
조인 조건으로 열 이름이 같을 필요는 없는데 비교를 위해 데이터 유형은 같아야 함

05-1 조인의 종류

▪ 내부 조인

▪ INNER JOIN 문으로 테이블 3개 이상 조인하기

- 테이블을 3개 이상 조인할 때는 두 테이블의 관계가 다대다 관계인 경우가 많음
- 그림을 통해 살펴보면 고객 정보, 고객 주소, 도시 이름의 데이터를 조회하기 위해서는 customer, address, city 테이블 3개를 조인해야 함



customer	
customer_id	고객 번호
store_id	스토어 번호
first_name	이름
last_name	성
email	이메일
address_id	주소 번호
active	활성화 유무
create_date	데이터 입력 시간
last_update	데이터 수정 시간

address	
address_id	주소 번호
address	상세 주소1
address2	상세 주소2
district	거리 이름
city_id	도시 번호
postal_code	우편번호
phone	전화번호
location	지역
last_update	데이터 수정 시간

city	
city_id	도시 번호
city	도시 이름
country_id	국가 번호
last_update	데이터 수정 시간

05-1 조인의 종류

▪ 내부 조인

▪ INNER JOIN 문으로 테이블 3개 이상 조인하기

- 테이블 2개를 조인하는 형식과 큰 차이는 없음

테이블 3개 이상일 때 INNER JOIN 문의 기본 형식

```
SELECT
        [열]
FROM [테이블 1]
        INNER JOIN [테이블 2] ON [테이블 1.열] = [테이블 2.열]
        INNER JOIN [테이블 3] ON [테이블 2.열] = [테이블 3.열]
WHERE [검색 조건]
```

05-1 조인의 종류

▪ 내부 조인

▪ INNER JOIN 문으로 테이블 3개 이상 조인하기

- first_name이 ROSA인 데이터의 고객 정보, 주소, 어느 도시인지를 조회하는 쿼리

Do it! 3개의 테이블을 조인한 테이블에서 조건에 맞는 데이터 조회

```
SELECT
        a.customer_id, a.first_name, a.last_name,
        b.address_id, b.address, b.district, b.postal_code,
        c.city_id, c.city
FROM customer AS a
        INNER JOIN address AS b ON a.address_id = b.address_id
        INNER JOIN city AS c ON b.city_id = c.city_id
WHERE a.first_name = 'ROSA';
```

실행 결과

	customer_id	first_name	last_name	address_id	address	district	postal_code	city_id	city
▶	112	ROSA	REYNOLDS	116	793 Cam Ranh Avenue	California	87057	292	Lancaster

05-1 조인의 종류

▪ 외부 조인

- 두 테이블을 조인해 둘 중의 한 테이블에만 있는 데이터를 조회해야 하는 경우에 외부 조인(OUTER JOIN)을 사용
- 예를 들어 상품을 주문한 고객과 상품을 주문하지 않은 고객 데이터를 모두 조회하고 싶다면 외부 조인을 사용할 수 있음
- 즉, 외부 조인은 열의 일치 여부를 고려하지 않고 한 쪽 테이블과 다른 쪽 테이블에 조인할 때 사용함

05-1 조인의 종류

▪ 외부 조인

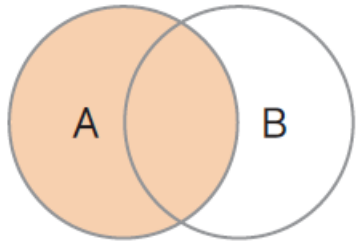
OUTER JOIN의 기본 형식

```
SELECT
        [열]
FROM [테이블 1]
        [LEFT | RIGHT | FULL] OUTER JOIN [테이블 2] ON [테이블 1.열] = [테이블 2.열]
WHERE [검색 조건]
```

- OUTER JOIN은 LEFT, RIGHT, FULL 중에서 한 옵션을 지정해야 함
- 이 옵션들은 기준이 되는 테이블을 정하는 것임
- 예를 들어 A, B 테이블이 좌우에 나란히 있다면
  A 테이블을 기준으로 B 테이블을 조인하고 싶다면 **LEFT**를 사용하고
  B 테이블을 기준으로 A 테이블을 조인하고 싶다면 **RIGHT**를 사용하면 됨

05-1 조인의 종류

- **외부 조인**
  - **LEFT OUTER JOIN으로 외부 조인하기**



LEFT OUTER JOIN 벤 다이어그램

05-1 조인의 종류

▪ 외부 조인

▪ LEFT OUTER JOIN으로 외부 조인하기

- LEFT OUTER JOIN을 하면 왼쪽에 놓인 고객 테이블이 우선 조회 결과(기준)에 포함됨
- 그런 다음 고객 테이블의 고객 번호와 주문 테이블의 고객 번호를 비교하여
  고객 테이블에 있는 고객 번호의 데이터를 결과에 포함시키고
  고객 번호 0003에 해당하는 데이터는 없으므로 이와 같이 NULL로 처리함
- 즉, LEFT(고객 테이블)의 데이터는 모두 표시된 상태임

고객 테이블

고객 번호	고객 이름
0001	한성우
0002	신지은
0003	이시연

주문 테이블

주문 번호	고객 번호	주문 날짜
100	0001	2024-02-16
101	0002	2024-02-17
102	0004	2024-02-22

LEFT OUTER JOIN ON 고객 번호

고객 번호	고객 이름	주문 번호	고객 번호	주문 날짜
0001	한성우	100	0001	2024-02-16
0002	신지은	101	0002	2024-02-17
0003	이시연	NULL	NULL	NULL

05-1 조인의 종류

▪ 외부 조인

▪ LEFT OUTER JOIN으로 외부 조인하기

Do it! LEFT OUTER JOIN 문을 적용한 예

```
SELECT
        [고객.고객 번호], [고객.고객 이름], [주문.주문 번호], [주문.고객 번호], [주문.주문 날짜]
FROM [고객]
        LEFT OUTER JOIN [주문] ON [고객.고객 번호] = [주문.고객 번호]
```

05-1 조인의 종류

▪ 외부 조인

▪ LEFT OUTER JOIN으로 외부 조인하기

1. 다음과 같이 쿼리를 입력해 address 테이블과 store 테이블을 address.id 열을 조인 조건으로 하여 LEFT OUTER JOIN해 보자

Do it! LEFT OUTER JOIN한 결과 조회

```
SELECT
        a.address, a.address_id AS a_address_id,
        b.address_id AS b_address_id, b.store_id
FROM address AS a
        LEFT OUTER JOIN store AS b ON a.address_id = b.address_id;
```

실행 결과

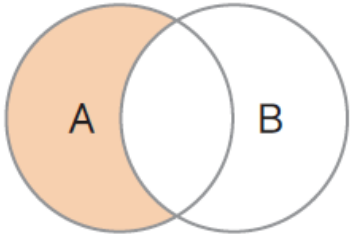
address	a_address_id	b_address_id	store_id
47 MySakila Drive	1	1	1
28 MySQL Boulevard	2	2	2
23 Workhaven Lane	3	NULL	NULL
1411 Lillydale Drive	4	NULL	NULL
1913 Hanoi Way	5	NULL	NULL
1121 Loja Avenue	6	NULL	NULL
692 Joliet Street	7	NULL	NULL
1566 Inegl Manor	8	NULL	NULL

05-1 조인의 종류

▪ 외부 조인

▪ LEFT OUTER JOIN으로 외부 조인하기

1. 실행 결과, 두 테이블의 address_id 열을 사용하여 address 테이블을 기준으로 외부 조인했기 때문에 store 테이블에 없는 address_id의 경우 NULL로 출력됨
2. 이번에는 다음 벤 다이어그램처럼 왼쪽에 있는 데이터, 즉 기준 테이블에 있는 데이터만 추출해 보자.
   그러려면 LEFT OUTER JOIN 결과에서 NULL이 있는 데이터만 골라내면 됨



LEFT OUTER JOIN에서 기준 테이블의 데이터만 추출한 벤 다이어그램

05-1 조인의 종류

▪ 외부 조인

▪ LEFT OUTER JOIN으로 외부 조인하기

2. 다음은 address 테이블을 기준으로 store 테이블과 LEFT OUTER JOIN한 다음, address 테이블에 있는 데이터만 조회한 쿼리임

Do it! LEFT OUTER JOIN으로 조회한 결과에서 NULL만 조회

```
SELECT
        a.address, a.address_id AS a_address_id,
        b.address_id AS b_address_id, b.store_id
FROM address AS a
        LEFT OUTER JOIN store AS b ON a.address_id = b.address_id
WHERE b.address_id IS NULL
```

실행 결과

address	a_address_id	b_address_id	store_id
23 Workhaven Lane	3	NULL	NULL
1411 Lillydale Drive	4	NULL	NULL
1913 Hanoi Way	5	NULL	NULL
1121 Loja Avenue	6	NULL	NULL
692 Joliet Street	7	NULL	NULL
1566 Inegl Manor	8	NULL	NULL
53 Idfu Parkway	9	NULL	NULL
1795 Santiago de Compostela Way	10	NULL	NULL

05-1 조인의 종류

▪ 외부 조인

▪ LEFT OUTER JOIN으로 외부 조인하기

2. OUTER JOIN한 결과에서 NULL을 조회하면
   이와 같이 기준 테이블(여기에서는 왼쪽)에만 존재하는 데이터를 조회할 때 사용할 수 있음

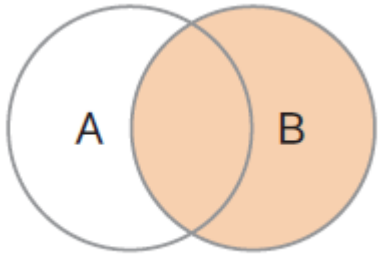
   실전에서는 A라는 상품과 B라는 상품이 있을 때,
   B 상품은 구매한 적이 없고 A 상품만 구매한 고객을 조회할 때 활용할 수 있음

05-1 조인의 종류

▪ 외부 조인

▪ RIGHT OUTER JOIN으로 외부 조인하기

- LEFT OUTER JOIN과 원리는 같고 기준 테이블만 다름



RIGHT OUTER JOIN 벤 다이어그램

고객 테이블	
고객 번호	고객 이름
0001	한성우
0002	신지은
0003	이시연

주문 테이블		
주문 번호	고객 번호	주문 날짜
100	0001	2024-02-16
101	0002	2024-02-17
102	0004	2024-02-22

RIGHT OUTER JOIN ON 고객 번호

고객 번호	고객 이름	주문 번호	고객 번호	주문 날짜
0001	한성우	100	0001	2024-02-16
0002	신지은	101	0002	2024-02-17
NULL	NULL	102	0004	2024-02-22

- 오른쪽의 주문 테이블의 모든 데이터가 포함되었고,
주문 테이블에서 고객 번호가 일치한 고객 테이블의 데이터만 표시됨
- 주문 테이블에는 있지만 고객 테이블에 없는 데이터는 NULL로 표시됨

05-1 조인의 종류

▪ 외부 조인

▪ RIGHT OUTER JOIN으로 외부 조인하기

1. 다음은 앞서 LEFT OUTER JOIN에 사용한 쿼리를 RIGHT OUTER JOIN으로 변경한 것임. 두 테이블의 address_id 열을 사용해 store 테이블을 기준으로 외부 조인함

Do it! RIGHT OUTER JOIN한 결과 조회

```
SELECT
        a.address, a.address_id AS a_address_id,
        b.address_id AS b_address_id, b.store_id
FROM address AS a
        RIGHT OUTER JOIN store AS b ON a.address_id = b.address_id;
```

실행 결과

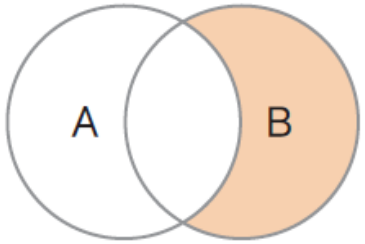
address	a_address_id	b_address_id	store_id
47 MySakila Drive	1	1	1
28 MySQL Boulevard	2	2	2

05-1 조인의 종류

▪ 외부 조인

▪ RIGHT OUTER JOIN으로 외부 조인하기

2. 이번에는 다음 벤 다이어그램처럼 오른쪽에 있는 데이터,
   즉 기준 테이블에 있는 데이터만 추출해 보자.
   그러려면 RIGHT OUTER JOIN 결과에서 NULL이 있는 데이터만 골라내면 됨



RIGHT OUTER JOIN에서 기준 테이블의 데이터만 추출한 벤 다이어그램

05-1 조인의 종류

▪ 외부 조인

▪ RIGHT OUTER JOIN으로 외부 조인하기

2. 여기서는 왼쪽에 store 테이블이, 오른쪽에 address 테이블이 있다고 가정함. address 테이블을 기준으로 store 테이블과 RIGHT OUTER JOIN한 다음, address 테이블에 있는 데이터만 조회해 보자.

Do it! RIGHT OUTER JOIN으로 조회한 결과에서 NULL만 조회

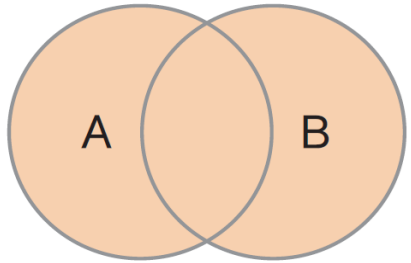
```
SELECT
        a.address_id AS a_address_id, a.store_id,
        b.address, b.address_id AS b_address_id
FROM store AS a
        RIGHT OUTER JOIN address AS b ON a.address_id = b.address_id
WHERE a.address_id IS NULL;
```

실행 결과

a_address_id	store_id	address	b_address_id
NULL	NULL	23 Workhaven Lane	3
NULL	NULL	1411 Lillydale Drive	4
NULL	NULL	1913 Hanoi Way	5
NULL	NULL	1121 Loja Avenue	6
NULL	NULL	692 Joliet Street	7
NULL	NULL	1566 Inegl Manor	8
NULL	NULL	53 Idfu Parkway	9
NULL	NULL	1795 Santiago de Compostela Way	10

05-1 조인의 종류

- 외부 조인
  - **FULL OUTER JOIN으로 외부 조인하기**
    - FULL OUTER JOIN은
      LEFT OUTER JOIN과 RIGHT OUTER JOIN을 합친 것임
    - 조인 조건에 일치하지 않는 항목, 일치하는 항목 모두 표시됨
    - 하지만 FULL OUTER JOIN은 실제 사용하는 경우는 드물다
    - 가끔 데이터베이스 디자인이나 데이터에 몇 가지 문제가 있어서 데이터의 누락이나 오류를 찾아낼 때 사용함
    - 예를 들어 잘못된 고객 번호로 주문 내역이 기록된 것은 없는지 확인하고 싶다면
      FULL OUTER JOIN이 적절함



FULL OUTER JOIN 벤 다이어그램

05-1 조인의 종류

▪ 외부 조인

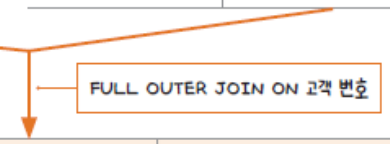
▪ FULL OUTER JOIN으로 외부 조인하기

고객 테이블

고객 번호	고객 이름
0001	한성우
0002	신지은
0003	이시연

주문 테이블

주문 번호	고객 번호	주문 날짜
100	0001	2024-02-16
101	0002	2024-02-17
102	0004	2024-02-22



고객 번호	고객 이름	주문 번호	고객 번호	주문 날짜
0001	한성우	100	0001	2024-02-16
0002	신지은	101	0002	2024-02-17
0003	이시연	NULL	NULL	NULL
NULL	NULL	102	0004	2024-02-22

05-1 조인의 종류

▪ 외부 조인

▪ FULL OUTER JOIN으로 외부 조인하기

1. 다른 DBMS의 경우 FULL OUTER JOIN 구문을 지원하지만
   MySQL에서는 FULL OUTER JOIN을 지원하지 않음

   그러므로 FULL OUTER JOIN 효과를 내려면
   LEFT OUTER JOIN의 결과와 RIGHT OUTER JOIN의 결과를 합쳐서(**UNION**)
   FULL OUTER JOIN처럼 구현함

05-1 조인의 종류

▪ 외부 조인

▪ FULL OUTER JOIN으로 외부 조인하기

Do it! FULL OUTER JOIN한 결과 조회

```
SELECT
        a.address_id AS a_address_id, a.store_id,
        b.address, b.address_id AS b_address_id
FROM store AS a
        LEFT OUTER JOIN address AS b ON a.address_id = b.address_id

UNION

SELECT
        a.address_id AS a_address_id, a.store_id,
        b.address, b.address_id AS b_address_id
FROM store AS a
        RIGHT OUTER JOIN address AS b ON a.address_id = b.address_id;
```

실행 결과

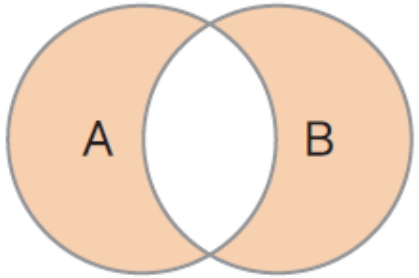
a_address_id	store_id	address	b_address_id
NULL	NULL	936 Salzburg Lane	82
NULL	NULL	586 Tete Way	83
NULL	NULL	1888 Kabul Drive	84
NULL	NULL	320 Baiyin Parkway	85
NULL	NULL	927 Baha Blanca Parkway	86
NULL	NULL	929 Tallahassee Loop	87
NULL	NULL	125 Citt del Vaticano Boulevard	88
NULL	NULL	1557 Ktahya Boulevard	89

05-1 조인의 종류

▪ 외부 조인

▪ FULL OUTER JOIN으로 외부 조인하기

2. 다음 벤 다이어그램과 같이
FULL OUTER JOIN을 사용하여
LEFT 테이블, RIGHT 테이블에 있는 테이터만 추출하려면
NULL 데이터를 조회하면 됨



FULL OUTER JOIN에서 양쪽 테이블의 데이터만 추출한 벤 다이어그램

05-1 조인의 종류

▪ 외부 조인

▪ FULL OUTER JOIN으로 외부 조인하기

Do it! FULL OUTER JOIN으로 조회한 결과에서 NULL만 조회

```
SELECT
        a.address_id AS a_address_id, a.store_id,
        b.address, b.address_id AS b_address_id
FROM store AS a
        LEFT OUTER JOIN address AS b ON a.address_id = b.address_id
WHERE b.address_id IS NULL

UNION

SELECT
        a.address_id AS a_address_id, a.store_id,
        b.address, b.address_id AS b_address_id
FROM store AS a
        RIGHT OUTER JOIN address AS b ON a.address_id = b.address_id
WHERE a.address_id IS NULL;
```

실행 결과

a_address_id	store_id	address	b_address_id
NULL	NULL	287 Cuautla Boulevard	436
NULL	NULL	1766 Almirante Brown Street	437
NULL	NULL	596 Huixquilucan Place	438
NULL	NULL	1351 Aparecida de Goinia Parkway	439
NULL	NULL	722 Bradford Lane	440
NULL	NULL	983 Santa F Way	441
NULL	NULL	1245 Ibirit Way	442
NULL	NULL	1836 Korla Parkway	443
NULL	NULL	231 Kaliningrad Place	444
NULL	NULL	495 Bhimavaram Lane	445

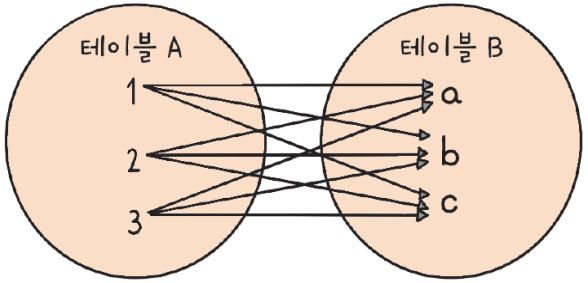
05-1 조인의 종류

▪ 교차 조인

- 자주 사용하지는 않지만
  각 테이블의 모든 경우의 수를 조합한 데이터가 필요할 경우
  교차 조인(CROSS JOIN)을 사용할 수 있음
- 교차 조인은 카르테시안(cartesian) 곱이라고도 함

CROSS JOIN의 기본 형식

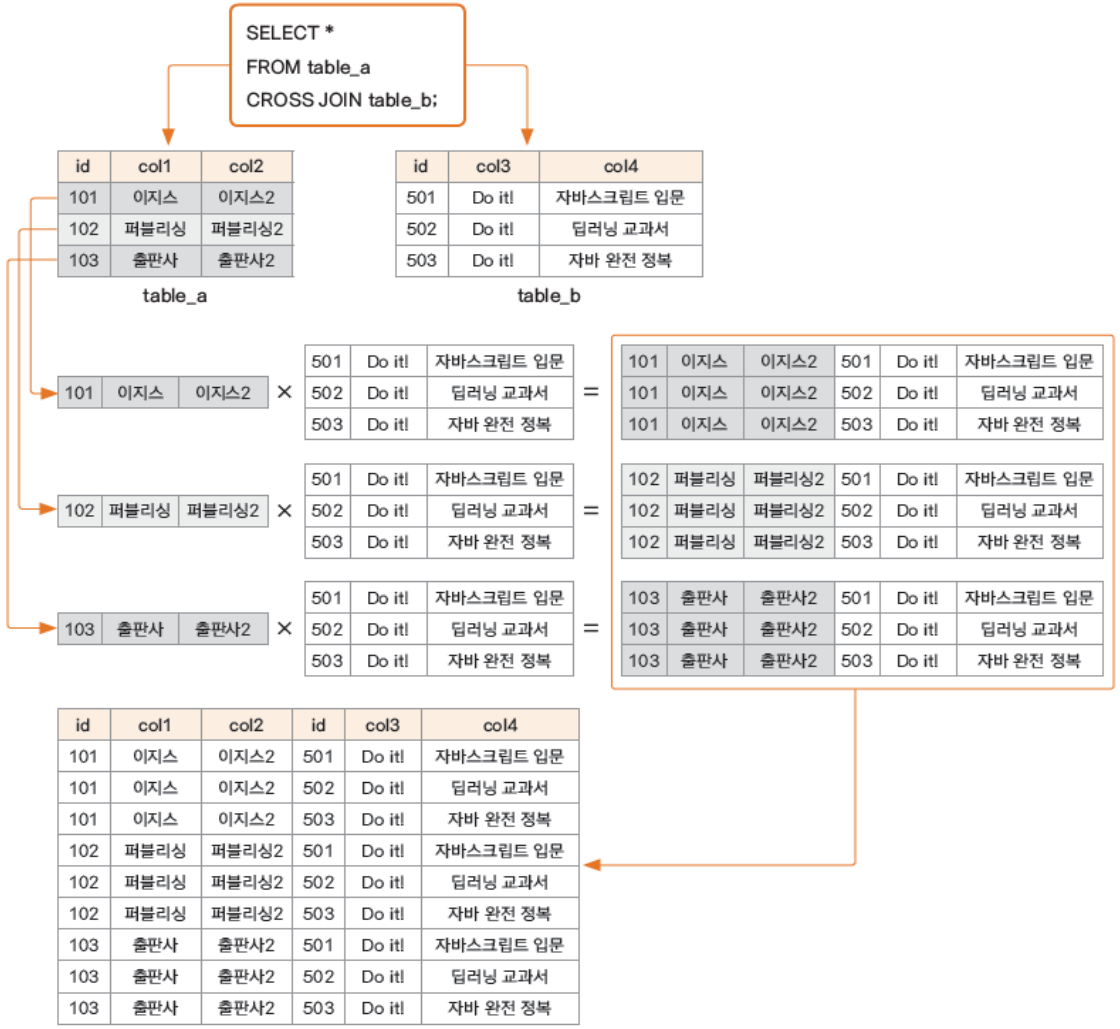
```
SELECT [열]
FROM [테이블 1]
        CROSS JOIN [테이블 2]
WHERE [검색 조건]
```



교차 조인

05-1 조인의 종류

▪ 교차 조인



id	col1	col2
101	이지스	이지스2
102	퍼블리싱	퍼블리싱2
103	출판사	출판사2

id	col3	col4
501	Do it!	자바스크립트 입문
502	Do it!	딥러닝 교과서
503	Do it!	자바 완전 정복

id	col1	col2	id	col3	col4
101	이지스	이지스2	501	Do it!	자바스크립트 입문
101	이지스	이지스2	502	Do it!	딥러닝 교과서
101	이지스	이지스2	503	Do it!	자바 완전 정복
102	퍼블리싱	퍼블리싱2	501	Do it!	자바스크립트 입문
102	퍼블리싱	퍼블리싱2	502	Do it!	딥러닝 교과서
102	퍼블리싱	퍼블리싱2	503	Do it!	자바 완전 정복
103	출판사	출판사2	501	Do it!	자바스크립트 입문
103	출판사	출판사2	502	Do it!	딥러닝 교과서
103	출판사	출판사2	503	Do it!	자바 완전 정복

05-1 조인의 종류

▪ 교차 조인

- 교차 조인은 사용할 일이 많지 않지만 샘플 데이터를 만들거나 각 행에 동일한 숫자의 데이터를 만들어야 할 때 활용할 수 있음

1. CROSS JOIN을 실습을 위해 먼저 샘플 데이터를 생성해 보자

Do it! 샘플 데이터 생성

```
CREATE TABLE doit_cross1(num INT);
CREATE TABLE doit_cross2(name VARCHAR(10));
INSERT INTO doit_cross1 VALUES (1), (2), (3);
INSERT INTO doit_cross2 VALUES ('Do'), ('It'), ('SQL');
```

05-1 조인의 종류

▪ 교차 조인

2. 샘플 데이터를 생성한 후 다음과 같이 CROSS JOIN 문을 작성하여 실행해 보자.

Do it! CROSS JOIN 문을 적용한 쿼리

```
SELECT
        a.num, b.name
FROM doit_cross1 AS a
        CROSS JOIN doit_cross2 AS b
ORDER BY a.num;
```

실행 결과

num	name
1	Do
1	It
1	SQL
2	Do
2	It
2	SQL
3	Do
3	It
3	SQL

결과를 살펴보면 doit_cross1, doit_corss2 테이블의 데이터가
총 9개 생성된 것을 확인할 수 있음

05-1 조인의 종류

▪ 교차 조인

3. WHERE 문을 사용해 조건으로 num이 1인 데이터를 조회하는 쿼리도 진행해 보자.

Do it! WHERE 문을 사용한 CROSS JOIN

```
SELECT
        a.num, b.name
FROM doit_cross1 AS a
        CROSS JOIN doit_cross2 AS b
WHERE a.num = 1;
```

실행 결과

	num	name
▶	1	Do
	1	It
	1	SQL

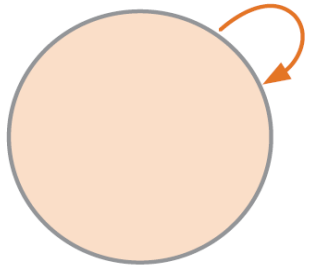
CROSS JOIN의 결과에서 doit_cross1 테이블의 num 열값이
1인 것만 출력되는 것을 확인할 수 있음

05-1 조인의 종류

▪ 셀프 조인

- 셀프 조인(SELF JOIN)은 동일한 테이블을 사용하는 특수한 조인임
- 내부 조인, 외부 조인, 교차 조인은 모두 2개 이상의 테이블을 조인한 반면 셀프 조인의 경우 자기 자신과 조인한다는 의미임

- 주의할 사항은 반드시 테이블의 별칭을 사용해야 한다.
  소스는 하나의 테이블이지만 2개의 테이블처럼 사용하기 때문에
  각 열에 서로 다른 별칭을 붙여 조인해야 함



셀프 조인

05-1 조인의 종류

▪ 셀프 조인

1. **Do it! SELF JOIN 문을 적용한 쿼리 1**

```
SELECT a.customer_id AS a_customer_id, b.customer_id AS b_customer_id
FROM customer AS a
        INNER JOIN customer AS b ON a.customer_id = b.customer_id
```

실행 결과

	a_customer_id	b_customer_id
▶	1	1
	2	2
	3	3
	5	5
	7	7
	10	10
	12	12
	15	15
	17	17
	19	19

05-1 조인의 종류

▪ 셀프 조인

2. 좀 더 실전에 가까운 예로 다른 쿼리도 작성해 보자.
   다음은 payment 테이블에서 전일 대비 수익이 얼마인지를 알아보는 쿼리임

Do it! SELF JOIN 문을 적용한 쿼리 2

```
SELECT
        a.payment_id, a.amount, b.payment_id, b.amount, b.amount - a.amount AS profit_amount
FROM payment AS a
        LEFT OUTER JOIN payment AS b ON a.payment_id = b.payment_id -1;
```

실행 결과

payment_id	amount	payment_id	amount	profit_amount
1	2.99	2	0.99	-2.00
2	0.99	3	5.99	5.00
3	5.99	4	0.99	-5.00
4	0.99	5	9.99	9.00
5	9.99	6	4.99	-5.00
6	4.99	7	4.99	0.00
7	4.99	8	0.99	-4.00
8	0.99	9	3.99	3.00
9	3.99	10	5.99	2.00
10	5.99	11	5.99	0.00

05-1 조인의 종류

▪ 셀프 조인

2. payment 테이블을 2번 사용하는 것이므로
a와 b를 활용한 별칭을 사용하여 테이블을 구분함.

ON 문을 살펴보면 a.payment_id = payment_id -1으로
왼쪽 테이블의 id와 오른쪽 테이블의 id -1의 ID를 나열했고,
a의 amount 가격과 b의 amount 가격 차이를 계산함.

이를 응용하면 전일과 오늘의 매출을 비교하여 수익 유무를 알 수 있음

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

- 서브 쿼리(subqueries)는 쿼리 안에 포함되어 있는 또 다른 쿼리를 말함
- 서브 쿼리는 조인하지 않은 상태에서 다른 테이블과 일치하는 행을 찾거나 조인 결과를 다시 조인할 때 사용할 수 있음

서브 쿼리의 특징

1. 서브 쿼리는 반드시 소괄호로 감싸 사용한다.
2. 서브 쿼리는 주 쿼리를 실행하기 전에 1번만 실행된다.
3. 비교 연산자와 함께 서브 쿼리를 사용하는 경우 서브 쿼리를 연산자 오른쪽에 기술해야 한다.
4. 서브 쿼리 내부에는 정렬 구문인 ORDER BY 문을 사용할 수 없다.

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

- 서브 쿼리 중에서 WHERE 문에 사용하는 서브 쿼리를 중첩 서브 쿼리(nested subquery)라고 함
- 중첩 서브 쿼리는 WHERE 다음에 오는 조건문의 일부로 사용함
- 서브 쿼리를 비교 연산자와 함께 사용할 때는 반드시 서브 쿼리의 반환 결과가 1건이라도 있어야 함
- 만약 서브 쿼리의 반환 결과가 2건 이상인 경우에는 비교 연산자가 아닌 다중 행 연산자를 사용해야 함

연산자	설명
IN	서브 쿼리의 결과에 존재하는 임의의 값과 동일한 조건 검색
ALL	서브 쿼리의 결과에 존재하는 모든 값을 만족하는 조건 검색
ANY	서브 쿼리의 결과에 존재하는 어느 하나의 값이라도 만족하는 조건 검색
EXISTS	서브 쿼리의 결과를 만족하는 값이 존재하는지 여부 확인

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 단일 행 서브 쿼리 사용하기

- 서브 쿼리의 결과로 1행만 반환되는 쿼리

단일 행 서브 쿼리의 기본 형식

```
SELECT [열]
FROM [테이블]
WHERE [열] = (SELECT [열] FROM [테이블])
```

Do it! 단일 행 서브 쿼리 적용

```
SELECT * FROM customer
WHERE customer_id = (SELECT customer_id FROM customer WHERE first_name = 'ROSA');
```

실행 결과

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
▶	112	2	ROSA	REYNOLDS	ROSA.REYNOLDS@sakilacustomer.org	116	1	2006-02-14 22:04:36	2006-02-15 04:57:20

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 단일 행 서브 쿼리 사용하기

Do it! 잘못된 단일 행 서브 쿼리 적용 시 오류 발생 예

```
SELECT * FROM customer
WHERE customer_id = (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

- WHERE 문에 사용한 서브 쿼리가 여러 행을 반환하면 비교 연산자 규칙에 어긋나므로 오류가 발생함

```
Error Code: 1242. Subquery returns more than 1 row
```

- 서브 쿼리의 결괏값이 1행이 아니어서 오류가 발생했다는 내용을 담고 있음

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

- 다중 행 서브 쿼리란 서브 쿼리에서 결과로 2행 이상인 반환되는 경우를 말함

- 서브 쿼리가 다중 행을 반환하기 위해서는
  앞서 잠깐 살펴본 IN, ANY, EXISTS, ALL 등의 다중행 연산자를 활용함

연산자	설명
IN	서브 쿼리의 결과에 존재하는 임의의 값과 동일한 조건 검색
ALL	서브 쿼리의 결과에 존재하는 모든 값을 만족하는 조건 검색
ANY	서브 쿼리의 결과에 존재하는 어느 하나의 값이라도 만족하는 조건 검색
EXISTS	서브 쿼리의 결과를 만족하는 값이 존재하는지 여부 확인

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

1. 첫 번째로 IN 연산자를 활용한 다중 행 서브 쿼리를 알아보자.
   IN을 활용한 다중 행 서브 쿼리의 기본 형식은 다음과 같음.
   쿼리의 형태를 보면 WHERE 문에 있는 IN 문의 소괄호 안에 서브 쿼리를 작성함

IN을 활용한 다중 행 서브 쿼리의 기본 형식

```
SELECT [열]
FROM [테이블]
WHERE [열] IN (SELECT [열] FROM [테이블])
```

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

1. 두 쿼리의 실행 결과는 같음.

Do it! IN을 활용한 다중 행 서브 쿼리 적용 1

```
SELECT * FROM customer
WHERE first_name IN ('ROSA', 'ANA');
```

실행 결과

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
112	2	ROSA	REYNOLDS	ROSA.REYNOLDS@sakilacustomer.org	116	1	2006-02-14 22:04:36	2006-02-15 04:57:20
181	2	ANA	BRADLEY	ANA.BRADLEY@sakilacustomer.org	185	1	2006-02-14 22:04:36	2006-02-15 04:57:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Do it! IN을 활용한 다중 행 서브 쿼리 적용 2

```
SELECT * FROM customer
WHERE customer_id IN (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

2. 이번에도 결과는 같지만 다른 쿼리 형식으로 작성해 보자.

   테이블 3개를 조인하는 쿼리와 같은 결과를 내는 쿼리를
   IN을 활용한 서브 쿼리로도 작성해 보자.

   두 번째 쿼리를 살펴보면
   WHERE 문에 사용한 서브 쿼리 안에서
   테이블 2개를 조인한 결과 행을 IN의 소괄호 안에 작성함

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

2.

Do it! 테이블 3개를 조인하는 쿼리

```
SELECT
        a.film_id, a.title
FROM film AS a
        INNER JOIN film_category AS b ON a.film_id = b.film_id
        INNER JOIN category AS c ON b.category_id = c.category_id
WHERE c.name = 'Action';
```

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

2. **Do it! IN을 활용한 서브 쿼리 적용**

```
SELECT
        film_id, title
FROM film
WHERE film_id IN (
        SELECT a.film_id
        FROM film_category AS a
                INNER JOIN category AS b ON a.category_id = b.category_id
        WHERE b.name = 'Action');
```

실행 결과

	film_id	title
▶	19	AMADEUS HOLY
	21	AMERICAN CIRCUS
	29	ANTITRUST TOMATOES
	38	ARK RIDGEMONT
	56	BAREFOOT MANCHURIAN
	67	BERETS AGENT
	97	BRIDE INTRIGUE
	105	BULL SHAWSHANK
	111	CADDYSHACK JEDI
	115	CAMPUS REMEMBER
	126	CASUALTIES ENCINO

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

- **WHERE 문에 서브 쿼리 사용하기**
  - **다중 행 서브 쿼리 사용하기**

2. 두 가지 방식으로 작성된 쿼리는 같은 결과를 출력함.

   쿼리문 작성 방법에는 하나의 정답이 있는 것이 아님.

   다양한 코드 작성법이 있으며,
   여기서는 조인으로 나온 결과를 서브 쿼리로도 동일하게 할 수 있다는 것을
   알려 주기 위해 이와 같이 작성한 것임

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

3. 다음은 NOT IN으로 film이 Action이 아닌 행을 조회해 보자

Do it! NOT IN을 활용한 서브 쿼리 적용

```
SELECT
        film_id, title
FROM film
WHERE film_id NOT IN (
        SELECT a.film_id
        FROM film_category AS a
                INNER JOIN category AS b ON a.category_id = b.category_id
        WHERE b.name = 'Action');
```

실행 결과

	film_id	title
▶	1	ACADEMY DINOSAUR
	2	ACE GOLDFINGER
	3	ADAPTATION HOLES
	4	AFFAIR PREJUDICE
	5	AFRICAN EGG
	6	AGENT TRUMAN
	7	AIRPLANE SIERRA
	8	AIRPORT POLLOCK
	9	ALABAMA DEVIL
	10	ALADDIN CALENDAR
	11	ALAMO VIDEOTAPE

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

4. ANY 연산자로는 서브 쿼리 결과에서
   값이 하나라도 만족하는 조건으로 데이터를 조회할 수 있음.

Do it! = ANY를 활용한 서브 쿼리 적용

```
SELECT * FROM customer
WHERE customer_id = ANY (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

실행 결과

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
112	2	ROSA	REYNOLDS	ROSA.REYNOLDS@sakilacustomer.org	116	1	2006-02-14 22:04:36	2006-02-15 04:57:20
181	2	ANA	BRADLEY	ANA.BRADLEY@sakilacustomer.org	185	1	2006-02-14 22:04:36	2006-02-15 04:57:20
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

5. 다음은 서브 쿼리의 결과값보다 작은 값을 조건으로 하여 데이터(행)를 반환하는 쿼리임

Do it! < ANY를 활용한 서브 쿼리 적용

```
SELECT * FROM customer
WHERE customer_id < ANY (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

실행 결과

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-15 04:57:20
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

6. 다음은 서브 쿼리의 결과값보다 큰 값을 조건으로 하여 데이터(행)를 반환하는 쿼리임

Do it! > ANY를 활용한 서브 쿼리 적용

```
SELECT * FROM customer
WHERE customer_id > ANY (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

실행 결과

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
113	2	CINDY	FISHER	CINDY.FISHER@sakilacustomer.org	117	1	2006-02-14 22:04:36	2006-02-15 04:57:20
114	2	GRACE	ELLIS	GRACE.ELLIS@sakilacustomer.org	118	1	2006-02-14 22:04:36	2006-02-15 04:57:20
115	1	WENDY	HARRISON	WENDY.HARRISON@sakilacustomer.org	119	1	2006-02-14 22:04:36	2006-02-15 04:57:20
116	1	VICTORIA	GIBSON	VICTORIA.GIBSON@sakilacustomer.org	120	1	2006-02-14 22:04:36	2006-02-15 04:57:20
117	1	EDITH	MCDONALD	EDITH.MCDONALD@sakilacustomer.org	121	1	2006-02-14 22:04:36	2006-02-15 04:57:20
118	1	KIM	CRUZ	KIM.CRUZ@sakilacustomer.org	122	1	2006-02-14 22:04:36	2006-02-15 04:57:20
119	1	SHERRY	MARSHALL	SHERRY.MARSHALL@sakilacustomer.org	123	1	2006-02-14 22:04:36	2006-02-15 04:57:20
120	2	SYLVIA	ORTIZ	SYLVIA.ORTIZ@sakilacustomer.org	124	1	2006-02-14 22:04:36	2006-02-15 04:57:20
121	1	JOSEPHINE	GOMEZ	JOSEPHINE.GOMEZ@sakilacustomer.org	125	1	2006-02-14 22:04:36	2006-02-15 04:57:20
122	1	THELMA	MURRAY	THELMA.MURRAY@sakilacustomer.org	126	1	2006-02-14 22:04:36	2006-02-15 04:57:20
123	2	SHANNON	FREEMAN	SHANNON.FREEMAN@sakilacustomer.org	127	1	2006-02-14 22:04:36	2006-02-15 04:57:20

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

7. EXISTS 문은 서브 쿼리의 결괏값이 있는지 없는지를 확인해서 1행이라도 있으면 TRUE, 없으면 FALSE를 반환함.

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

7. WHERE 문에 EXISTS 문을 사용해
서브 쿼리의 결괏값이 1행이라도 있으면 TRUE가 되어 주 쿼리를 실행하고,
주 쿼리에 작성된 전체 데이터를 검색하는 쿼리임

Do it! EXISTS를 활용한 서브 쿼리 적용: TRUE를 반환하는 경우

```
SELECT * FROM customer
WHERE EXISTS (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

WHERE 문에 사용된 서브 쿼리의 결과가
2건이므로 TRUE로 판단해 주 쿼리가 실행

실행 결과

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-15 04:57:20
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

8. 서브 쿼리의 결괏값이 없으면 FALSE이므로 주 쿼리를 실행하지 않으며, 아무것도 나타나지 않음

Do it! EXISTS를 활용한 서브 쿼리 적용: FALSE를 반환하는 경우

```
SELECT * FROM customer
WHERE EXISTS (SELECT customer_id FROM customer WHERE first_name IN ('KANG'));
```

실행 결과

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

WHERE 문에 사용된 서브 쿼리에 결괏값이 없으므로
FALSE로 판단되어 주 쿼리가 실행되지 않아서 아무런 결과가 나타나지 않음

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

9. NOT EXISTS는 EXISTS와 반대로 동작함.
다음은 바로 앞서 실습한 쿼리를 반대로 적용한 것임

Do it! NOT EXISTS를 활용한 서브 쿼리 적용: TRUE를 반환하는 경우

```
SELECT * FROM customer
WHERE NOT EXISTS (SELECT customer_id FROM customer WHERE first_name IN ('KANG'));
```

WHERE 문에 사용된 서브 쿼리에
결괏값이 없기 때문에 반대인 TRUE로
판단되어 주 쿼리가 실행되고
customer 테이블 전체가 출력

실행 결과

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20
6	2	JENNIFER	DAVIS	JENNIFER.DAVIS@sakilacustomer.org	10	1	2006-02-14 22:04:36	2006-02-15 04:57:20
7	1	MARIA	MILLER	MARIA.MILLER@sakilacustomer.org	11	1	2006-02-14 22:04:36	2006-02-15 04:57:20
8	2	SUSAN	WILSON	SUSAN.WILSON@sakilacustomer.org	12	1	2006-02-14 22:04:36	2006-02-15 04:57:20
9	2	MARGARET	MOORE	MARGARET.MOORE@sakilacustomer.org	13	1	2006-02-14 22:04:36	2006-02-15 04:57:20
10	1	DOROTHY	TAYLOR	DOROTHY.TAYLOR@sakilacustomer.org	14	1	2006-02-14 22:04:36	2006-02-15 04:57:20
11	2	LISA	ANDERSON	LISA.ANDERSON@sakilacustomer.org	15	1	2006-02-14 22:04:36	2006-02-15 04:57:20

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ WHERE 문에 서브 쿼리 사용하기

▪ 다중 행 서브 쿼리 사용하기

10. ALL 문은 서브 쿼리와 자주 사용하지는 않지만.
ALL 문은 서브 쿼리 결괏값에 있는 모든 값을 만족하는 조건을 주 쿼리에서 검색하여 결과를 반환함.

Do it! ALL을 활용한 서브 쿼리 적용

```
SELECT * FROM customer
WHERE customer_id = ALL (SELECT customer_id FROM customer WHERE first_name IN ('ROSA', 'ANA'));
```

실행 결과

	customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

서브 쿼리의 결괏값 모두를 만족하는 결과가 주 쿼리 결괏값에 없어 아무것도 나오지 않음
즉, WHERE customer_id = 112 AND customer_id = 181인 조건과 같음

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ FROM 문에 서브 쿼리 사용하기

- FROM 문에 사용한 서브 쿼리 결과는
  테이블처럼 사용되어 다른 테이블과 다시 조인할 수 있음.
  즉, 쿼리를 논리적으로 격리할 수 있음

- FROM 문에 사용하는 서브 쿼리의 기본 형식은 다음과 같음.
  이때, FROM 문에 사용하는 서브 쿼리는 인라인뷰(inline view)라고 함

FROM 문에 사용하는 서브 쿼리의 기본 형식

```
SELECT
        [열]
FROM [테이블] AS a
        INNER JOIN (SELECT [열] FROM [테이블] WHERE [열] = [값]) AS b ON [a.열] = [b.열]
WHERE [열] = [값]
```

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ FROM 문에 서브 쿼리 사용하기

- 쉽게 비교하기 위해 우선은 INNER JOIN 문으로 구성된 쿼리를 작성해 보자

Do it! 테이블 조인

```
SELECT
        a.film_id, a.title, a.special_features, c.name
FROM film AS a
        INNER JOIN film_category AS b ON a.film_id = b.film_id
        INNER JOIN category AS c ON b.category_id = c.category_id
WHERE a.film_id > 10 AND a.film_id < 20;
```

실행 결과

film_id	title	special_features	name
11	ALAMO VIDEOTAPE	Commentaries,Behind the Scenes	Foreign
12	ALASKA PHANTOM	Commentaries,Deleted Scenes	Music
13	ALI FOREVER	Deleted Scenes,Behind the Scenes	Horror
14	ALICE FANTASIA	Trailers,Deleted Scenes,Behind the Scenes	Classics
15	ALIEN CENTER	Trailers,Commentaries,Behind the Scenes	Foreign
16	ALLEY EVOLUTION	Trailers,Commentaries	Foreign
17	ALONE TRIP	Trailers,Behind the Scenes	Music
18	ALTER VICTORY	Trailers,Behind the Scenes	Animation
19	AMADEUS HOLY	Commentaries,Deleted Scenes,Behind the Scenes	Action

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ FROM 문에 서브 쿼리 사용하기

- 이번에는 조인을 서브 쿼리로 작성해 보자
  서브 쿼리의 결과가 마치 테이블처럼 동작하여 다시 다른 테이블과 조인하는 방식으로 처리됨

Do it! FROM 문에 서브 쿼리 적용

```
SELECT
        a.film_id, a.title, a.special_features, x.name
FROM film AS a
        INNER JOIN (
        SELECT
                b.film_id, c.name
        FROM film_category AS b
                INNER JOIN category AS c ON b.category_id = c.category_id
        WHERE b.film_id > 10 AND b.film_id < 20) AS x ON a.film_id = x.film_id;
```

실행 결과

	film_id	title	special_features	name
▶	11	ALAMO VIDEOTAPE	Commentaries,Behind the Scenes	Foreign
	12	ALASKA PHANTOM	Commentaries,Deleted Scenes	Music
	13	ALI FOREVER	Deleted Scenes,Behind the Scenes	Horror
	14	ALICE FANTASIA	Trailers,Deleted Scenes,Behind the Scenes	Classics
	15	ALIEN CENTER	Trailers,Commentaries,Behind the Scenes	Foreign
	16	ALLEY EVOLUTION	Trailers,Commentaries	Foreign
	17	ALONE TRIP	Trailers,Behind the Scenes	Music
	18	ALTER VICTORY	Trailers,Behind the Scenes	Animation
	19	AMADEUS HOLY	Commentaries,Deleted Scenes,Behind the Scenes	Action

두 쿼리문의 결과는 동일함.

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ SELECT 문에 서브 쿼리 사용하기

- SELECT 문에 사용한 서브 쿼리는 스칼라 서브 쿼리(scalar subqueries)라고 하며, 스칼라 서브 쿼리는 1개 이상 사용할 수 있음

스칼라 서브 쿼리의 기본 형식

```
SELECT
        [열], (SELECT <집계 함수> [열] FROM [테이블 2]
        WHERE [테이블 2.열] = [테이블 1.열]) AS a
FROM [테이블 1]
WHERE [조건]
```

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ SELECT 문에 서브 쿼리 사용하기

- 다음 쿼리는 조인으로 작성된 구문을 SELECT 서브 쿼리로 작성함
- 우선 조인으로 쿼리를 작성하여 결과를 확인하자

Do it! 테이블 조인

```
SELECT
        a.film_id, a.title, a.special_features, c.name
FROM film AS a
        INNER JOIN film_category AS b ON a.film_id = b.film_id
        INNER JOIN category AS c ON b.category_id = c.category_id
WHERE a.film_id > 10 AND a.film_id < 20;
```

실행 결과

film_id	title	special_features	name
11	ALAMO VIDEOTAPE	Commentaries,Behind the Scenes	Foreign
12	ALASKA PHANTOM	Commentaries,Deleted Scenes	Music
13	ALI FOREVER	Deleted Scenes,Behind the Scenes	Horror
14	ALICE FANTASIA	Trailers,Deleted Scenes,Behind the Scenes	Classics
15	ALIEN CENTER	Trailers,Commentaries,Behind the Scenes	Foreign
16	ALLEY EVOLUTION	Trailers,Commentaries	Foreign
17	ALONE TRIP	Trailers,Behind the Scenes	Music
18	ALTER VICTORY	Trailers,Behind the Scenes	Animation
19	AMADEUS HOLY	Commentaries,Deleted Scenes,Behind the Scenes	Action

05-2 쿼리 안에 또 다른 쿼리, 서브 쿼리

▪ SELECT 문에 서브 쿼리 사용하기

- 이번에는 SELECT 문에 서브 쿼리를 사용함

Do it! SELECT 문에 서브 쿼리 적용

```
SELECT
        a.film_id, a.title, a.special_features,
(SELECT c.name FROM film_category as b
INNER JOIN category AS c
ON b.category_id = c.category_id
WHERE a.film_id = b.film_id) AS name
FROM film AS a
WHERE a.film_id > 10 AND a.film_id < 20;
```

실행 결과

film_id	title	special_features	name
11	ALAMO VIDEOTAPE	Commentaries,Behind the Scenes	Foreign
12	ALASKA PHANTOM	Commentaries,Deleted Scenes	Music
13	ALI FOREVER	Deleted Scenes,Behind the Scenes	Horror
14	ALICE FANTASIA	Trailers,Deleted Scenes,Behind the Scenes	Classics
15	ALIEN CENTER	Trailers,Commentaries,Behind the Scenes	Foreign
16	ALLEY EVOLUTION	Trailers,Commentaries	Foreign
17	ALONE TRIP	Trailers,Behind the Scenes	Music
18	ALTER VICTORY	Trailers,Behind the Scenes	Animation
19	AMADEUS HOLY	Commentaries,Deleted Scenes,Behind the Scenes	Action

이 쿼리로도 동일한 결과를 출력함

05-3 공통 테이블 표현식이란?

- 공통 테이블 표현식(Common Table Expression, CTE)은
  실제 데이터베이스에 생성되는 테이블은 아니지만
  쿼리 실행 결과를 테이블처럼 활용하기 위한 논리적인 테이블을 만들 때 활용함

- 주로 데이터베이스에 없는 테이블이 필요할 때 사용하며,
  바로 다음에 실행할 SELECT 문에만
  이러한 공통 테이블 표현식을 사용해 데이터를 조회할 수 있음

- 공통 테이블 표현식은
  목적에 따라 일반 공통 테이블 표현식과
  결과를 재사용하는 재귀 공통 테이블 표현식으로 나뉨

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

- 쿼리 실행 결과인 데이터 집합을 하나의 테이블처럼 사용함

일반 CTE의 기본 형식

```
WITH [테이블] (열 1, 열 2, ...)
AS
(
        <SELECT 문>
)
SELECT [열] FROM [테이블];
```

- WITH 문의 [테이블](열 1, 열 2, ...)은 이 CTE 테이블을 구성하는 테이블 정보(테이블 이름, 구성할 열의 이름)를 담고 있음
- AS 이하엔 <SELECT 문>을 입력하여 CTE 테이블에 들어갈 데이터들을 조회함
- 일반 CTE는 복잡한 쿼리를 단순하게 만들 때 유용하게 사용됨

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

1. CTE 내부에서 조회한 데이터 집합을 CTE 외부의 SELECT 문에서 테이블처럼 참조해서 사용

Do it! 일반 CTE로 데이터 조회

```
WITH cte_customer (customer_id, first_name, email)
AS
(
        SELECT customer_id, first_name, email FROM customer
        WHERE customer_id >= 10 AND customer_id < 100
)
SELECT * FROM cte_customer;
```

실행 결과

customer_id	first_name	email
10	DOROTHY	DOROTHY.TAYLOR@sakilacustomer.org
11	LISA	LISA.ANDERSON@sakilacustomer.org
12	NANCY	NANCY.THOMAS@sakilacustomer.org
13	KAREN	KAREN.JACKSON@sakilacustomer.org
14	BETTY	BETTY.WHITE@sakilacustomer.org
15	HELEN	HELEN.HARRIS@sakilacustomer.org
16	SANDRA	SANDRA.MARTIN@sakilacustomer.org
17	DONNA	DONNA.THOMPSON@sakilacustomer.org
18	CAROL	CAROL.GARCIA@sakilacustomer.org
19	RUTH	RUTH.MARTINEZ@sakilacustomer.org
20	SHARON	SHARON.ROBINSON@sakilacustomer.org
21	MICHELLE	MICHELLE.CLARK@sakilacustomer.org
22	LAURA	LAURA.RODRIGUEZ@sakilacustomer.org

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

2. CTE에서 정의한 열과 CTE 내부의 SELECT 문의 열 목록이 다르면 오류가 발생

Do it! 일반 CTE에서 열 불일치로 인한 오류 발생 예

```
WITH cte_customer (customer_id, first_name, email)
AS
(
        SELECT customer_id, first_name, last_name, email FROM customer
        WHERE customer_id >= 10 AND customer_id < 100
)

SELECT * FROM cte_customer;
```

```
Error Code: 1353. In definition of view, derived table or common table expression,
SELECT list and column names list have different column counts
```

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

▪ UNION 문으로 CTE 결합해 보기

Do it! UNION ALL로 CTE 결합

```
WITH cte_customer (customer_id, first_name, email)
AS
(
        SELECT customer_id, first_name, email FROM customer WHERE
                customer_id >= 10    AND customer_id <= 15
        UNION ALL
        SELECT customer_id, first_name, email FROM customer WHERE
                customer_id >= 25    AND customer_id <= 30
)
SELECT * FROM cte_customer;
```

실행 결과

customer_id	first_name	email
10	DOROTHY	DOROTHY.TAYLOR@sakilacustomer.org
11	LISA	LISA.ANDERSON@sakilacustomer.org
12	NANCY	NANCY.THOMAS@sakilacustomer.org
13	KAREN	KAREN.JACKSON@sakilacustomer.org
14	BETTY	BETTY.WHITE@sakilacustomer.org
15	HELEN	HELEN.HARRIS@sakilacustomer.org
25	DEBORAH	DEBORAH.WALKER@sakilacustomer.org
26	JESSICA	JESSICA.HALL@sakilacustomer.org
27	SHIRLEY	SHIRLEY.ALLEN@sakilacustomer.org
28	CYNTHIA	CYNTHIA.YOUNG@sakilacustomer.org
29	ANGELA	ANGELA.HERNANDEZ@sakilacustomer....
30	MELISSA	MELISSA.KING@sakilacustomer.org

05-3 공통 테이블 표현식이란?

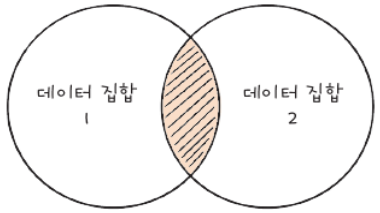
- **일반 CTE**
  - **UNION 문으로 CTE 결합해 보기**
    - UNION 문과 UNION ALL 문의 차이점은 중복을 제거한 행 포함 여부
    - customer_id <= 15 -> customer_id <= 27 로 바꾸어 UNION, UNION ALL 의 차이점을 확인할 수 있다.

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

▪ INTERSECT 문으로 CTE 결합해 보기

- 각 쿼리에서 반환한 결과에서 중복값을 걸러 반환



INTERSECT 문으로 CTE를 결합한 벤 다이어그램

Do it! INTERSECT 문으로 CTE 결합

```
WITH cte_customer (customer_id, first_name, email)
AS
(
        SELECT customer_id, first_name, email FROM customer WHERE customer_id >= 10
        AND customer_id <= 15
        INTERSECT
        SELECT customer_id, first_name, email FROM customer WHERE customer_id >= 12
        AND customer_id <= 20
)
SELECT * FROM cte_customer;
```

실행 결과

customer_id	first_name	email
12	NANCY	NANCY.THOMAS@sakilacustomer.org
13	KAREN	KAREN.JACKSON@sakilacustomer.org
14	BETTY	BETTY.WHITE@sakilacustomer.org
15	HELEN	HELEN.HARRIS@sakilacustomer.org

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

▪ EXCEPT 문으로 CTE 결합해 보기

Do it! EXCEPT 문으로 CTE 결합 1

```
WITH cte_customer (customer_id, first_name, email)
AS
(
        SELECT customer_id, first_name, email FROM customer WHERE customer_id >= 10
        AND customer_id <= 15
        EXCEPT
        SELECT customer_id, first_name, email FROM customer WHERE customer_id >= 12
        AND customer_id <= 20
)
SELECT * FROM cte_customer;
```

실행 결과

	customer_id	first_name	email
▶	10	DOROTHY	DOROTHY.TAYLOR@sakilacustomer.org
	11	LISA	LISA.ANDERSON@sakilacustomer.org

05-3 공통 테이블 표현식이란?

▪ 일반 CTE

▪ EXCEPT 문으로 CTE 결합해 보기

Do it! EXCEPT 문으로 CTE 결합 2

```
WITH cte_customer (customer_id, first_name, email)
AS
(
        SELECT customer_id, first_name, email FROM customer WHERE customer_id >= 12
        AND customer_id <= 20
        EXCEPT
        SELECT customer_id, first_name, email FROM customer WHERE customer_id >= 10
        AND customer_id <= 15
)
SELECT * FROM cte_customer;
```

실행 결과

customer_id	first_name	email
16	SANDRA	SANDRA.MARTIN@sakilacustomer.org
17	DONNA	DONNA.THOMPSON@sakilacustomer.org
18	CAROL	CAROL.GARCIA@sakilacustomer.org
19	RUTH	RUTH.MARTINEZ@sakilacustomer.org
20	SHARON	SHARON.ROBINSON@sakilacustomer.org

- EXCEPT 문으로 SELECT 문을 결합할 때 SELECT 문의 순서에 따라 그 결과가 달라짐을 알 수 있음

05-3 공통 테이블 표현식이란?

▪ 재귀 CTE

- 내부에서 함수가 자기 자신을 또다시 호출하는 것. 재귀 CTE는 CTE 결과를 CTE 내부의 쿼리에서 재사용함으로써 반복 실행하는 쿼리 구조를 갖음
- 재귀 CTE는 주로 조직도와 같은 계층 데이터를 검색할 때 많이 사용함

재귀 CTE의 기본 형식

```
WITH [CTE_테이블](열 1, 열 2, ...)

AS(
        <SELECT * FROM 테이블 A>
        UNION ALL
        <SELECT * FROM 테이블 B JOIN CTE_테이블>
)
SELECT * FROM [CTE_테이블];
```

05-3 공통 테이블 표현식이란?

▪ 재귀 CTE

- 피보나치 수열을 생성하는 쿼리
- 피보나치 수열은 숫자 0과 1(또는 1과 1)로 시작하며
  두 숫자를 합친 값이 그 다음 숫자가 되어 이어짐(0, 1, 1, 2, 3, 5, ...)

Do it! 재귀 CTE로 피보나치 수열 생성

```
WITH RECURSIVE fibonacci_number (n, fibonacci_n, next_fibonacci_n)
AS(
        SELECT 1, 0, 1
        UNION ALL
        SELECT n + 1, next_fibonacci_n, fibonacci_n + next_fibonacci_n
                FROM fibonacci_number WHERE n < 20
)
SELECT * FROM fibonacci_number;
```

실행 결과

	n	fibonacci_n	next_fibonacci_n
	1	0	1
	2	1	1
	3	1	2
	4	2	3
	5	3	5
	6	5	8
	7	8	13
	8	13	21
	9	21	34
	10	34	55