EXISTS 문법

현업 사용 정도 : ★ ★ ★ ★ ★

1. EXISTS 문법 알아보기
2. EXISTS 원리와 사용이유
3. 실습 문제 풀이

강사 강태우

1. EXISTS 문법 알아보기

EXISTS 문법의 이해를 위해 아래 테이블에서

회원 중에 연락처 정보가 존재하는 회원들만 찾아봅시다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

EXISTS 문법이란?

특정 조건을 만족하는 데이터가 존재하는지(EXISTS) 여부를 확인할 때 사용하는 문법입니다.

대표적인 상관서브쿼리 기술이며 현업에서 자주 사용하는 스킬입니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

연락처가 존재하는(조건)

직원을 찾는 기술

1. EXISTS 문법 알아보기

EXISTS 문법 작성 방법

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE EXISTS (
                SELECT 1
                  FROM TB_MEMBER_TEL
                 WHERE MEMBER_ID = A.MEMBER_ID
               ) ;
```

1. WHERE 에 사용할 때 특정 컬럼을 이용하지 않습니다.
2. SELECT 뒤에 숫자 1은 의미가 없습니다. 'X'도 가능하며 단순 문법 맞추기용도 입니다.
3. 메인쿼리의 컬럼 값을 빌려오고 있는 상관서브쿼리입니다. (A.MEMBER_ID)

EXISTS 원리 및 사용이유

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE EXISTS (
                SELECT 1
                  FROM TB_MEMBER_TEL     AAAAA
                 WHERE MEMBER_ID = A.MEMBER_ID
               ) ;
```

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A

조건에 일치하는 대상을 찾았습니다!
회원(AAAAA) 에 부합하는 조건이 있습니다.

이후의 행은 비교하지 않습니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

EXISTS 원리 및 사용이유

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE EXISTS (
                SELECT 1
                  FROM TB_MEMBER_TEL      BBBBB
                 WHERE MEMBER_ID = A.MEMBER_ID
               ) ;
```

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B

조건에 일치하는 대상을 찾았습니다!
회원(BBBBB) 에 부합하는 조건이 있습니다.

이후의 행은 비교하지 않습니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

EXISTS 원리 및 사용이유

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE MEMBER_ID IN (
                     SELECT MEMBER_ID
                       FROM TB_MEMBER_TEL
                     ) ;
```

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B

IN 과 서브쿼리를 이용해도 똑같은
결과를 출력할 수 있습니다.

하지만 성능은 EXISTS 가 훨씬 좋습니다.

IN을 이용했을 때 문제점

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE MEMBER_ID IN (
                     SELECT MEMBER_ID
                       FROM TB_MEMBER_TEL
                     ) ;
```

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B

조건에 일치하는 대상을 찾았습니다!
하지만 데이터를 끝까지 모두 비교합니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

2. EXISTS 원리와 사용이유

NOT EXISTS 문법의 이해를 위해 아래 테이블에서

회원 중에 연락처 정보가 존재하지 않는 회원들만 찾아봅시다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

NOT EXISTS 문법은 반대로
존재하지 않는 조건을 찾는 문법입니다.

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_MEMBER_TEL
                    WHERE MEMBER_ID = A.MEMBER_ID
                   );
```

MEMBER_ID	MEMBER_NAME
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

NOT EXISTS 원리 및 사용이유

(EXISTS 와 동일)

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_MEMBER_TEL
                    WHERE MEMBER_ID = A.MEMBER_ID
                   );
```

AAAAA

조건에 일치하는 대상을 찾았습니다!
회원(AAAAA) 에 부합하는 조건이 있습니다.
그러므로 출력하지 않습니다

이후의 행은 비교하지 않습니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

NOT EXISTS 원리 및 사용이유

(EXISTS 와 동일)

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_MEMBER_TEL
                    WHERE MEMBER_ID = A.MEMBER_ID
                   );
```

BBBBB

조건에 일치하는 대상을 찾았습니다!
회원(BBBBB) 에 부합하는 조건이 있습니다.
그러므로 출력하지 않습니다

이후의 행은 비교하지 않습니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

NOT EXISTS 원리 및 사용이유

(EXISTS 와 동일)

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_MEMBER_TEL
                    WHERE MEMBER_ID = A.MEMBER_ID
                   );
```

CCCCC

MEMBER_ID	MEMBER_NAME
CCCCC	사용자C

조건에 일치하는 대상이 없습니다!
그러므로 해당 데이터는 출력합니다.

이후의 행은 비교하지 않습니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

NOT EXISTS 원리 및 사용이유

(EXISTS 와 동일)

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_MEMBER_TEL
                    WHERE MEMBER_ID = A.MEMBER_ID
                   );
```

IIIII

MEMBER_ID	MEMBER_NAME
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

조건에 일치하는 대상이 없습니다!
그러므로 해당 데이터는 출력합니다.

이후의 행은 비교하지 않습니다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME
AAAAA	사용자A
BBBBB	사용자B
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

TB_MEMBER_TEL

MEMBER_ID	TEL_DV_CD	TEL_NO
AAAAA	집	062-123-1234
AAAAA	휴대폰	010-1231-1231
AAAAA	회사	02-9999-9999
BBBBB	집	062-555-7777
BBBBB	휴대폰	010-5555-8888

NOT EXISTS 원리 및 사용이유

(EXISTS 와 동일)

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE MEMBER_ID NOT IN (
                    SELECT MEMBER_ID
                      FROM TB_MEMBER_TEL
                  ) ;
```

MEMBER_ID	MEMBER_NAME
CCCCC	사용자C
DDDDD	사용자D
EEEEE	사용자E
FFFFF	사용자F
GGGGG	사용자G
HHHHH	사용자H
IIIII	사용자I

NOT IN 과 서브쿼리를 이용해도 똑같은 결과를 출력할 수 있습니다.

하지만 성능은 NOT EXISTS 가 훨씬 좋습니다.

그리고 치명적인 단점이 있습니다.

NOT IN을 이용했을 때 문제점

(만약 서브쿼리쪽에 NULL이 있었다면?)

```
SELECT MEMBER_ID
     , MEMBER_NAME
  FROM TB_MEMBER A
 WHERE MEMBER_ID NOT IN (
                         SELECT MEMBER_ID
                           FROM TB_MEMBER_TEL
                         UNION ALL

                         SELECT NULL
                           FROM DUAL --임의로 NULL 데이터를 추가
                        ) ;
```

MEMBER_ID
AAAAA
AAAAA
AAAAA
BBBBB
BBBBB
(null)

NOT IN 은 NULL 데이터가 포함되면

아무것도 출력하지 않습니다.

(NOT EXISTS 는 가능함)

3. 실습 문제 풀이

문제1) TB_ORDER 테이블은 주문 데이터를 입력받고 있습니다.

TB_ORDER 테이블의 PRD_ID 컬럼과 TB_PRD 테이블의 PRD_ID 컬럼을 활용하여

한번이라도 주문이 된 적이 있는 상품의 PRD_ID와 PRD_NAME 을 출력해주세요.

[힌트 : EXISTS 활용 , 한번이라도 주문이 되었다는 것은 TB_ORDER 테이블에 해당 값이 존재한다는 의미]

PRD_ID	PRD_NAME
P0002	에어컨
P0003	세탁기
P0004	건조기
P0020	수건

문제2) TB_MEMBER 테이블과 TB_ORDER 테이블을 활용하여 주문을 아직 한번도 하지 않는 회원이면서

GRADE_CD(등급코드) 가 3 인 회원의 MEMBER_ID , MEMBER_NAME , AGE 를 출력해주세요.

[힌트: NOT EXISTS 활용 , 한번도 주문을 안했다는 의미는 TB_ORDER 테이블에 해당 값이 없다는 의미 , 조건에서 AND 조건을 잘 사용해보세요]

MEMBER_ID	MEMBER_NAME	AGE	GRADE_CD
DDDDD	사용자D	30	3
FFFFF	사용자F	35	3

3. 실습 문제 풀이

문제3) [심화]

TB_MEMBER_LIKE 테이블은 회원이 어떤 상품 타입(PRD_TYPE)을 선호하는지 정보를 저장한 테이블입니다.

예를 들어 회원 BBBBB 는 선호하는 상품타입이 '가전' 과 '스마트폰' 입니다. (SELECT * FROM TB_MEMBER_LIKE ;)

이 때, TB_PRD (상품) 테이블에는 여러가지 상품 타입이 존재합니다.

BBBBB 회원이 선호하는 상품타입 외에 나머지 상품타입들을 아래와 같이 출력해주세요.

힌트)

```
SELECT [          ] PRD_TYPE
  FROM TB_PRD A
 WHERE [          ] (
                      SELECT 1
                        FROM TB_MEMBER_LIKE
                       WHERE [                    ]
                         AND MEMBER_ID = 'BBBBB'
                      );
```

PRD_TYPE
주방용품
가전
욕실용품

3. 실습 문제 풀이 (답)

문제1) TB_ORDER 테이블은 주문 데이터를 입력받고 있습니다.

TB_ORDER 테이블의 PRD_ID 컬럼과 TB_PRD 테이블의 PRD_ID 컬럼을 활용하여 한번이라도 주문이 된 적이 있는 상품의 PRD_ID와 PRD_NAME 을 출력해주세요.

[힌트 : EXISTS 활용 , 한번이라도 주문이 되었다는 것은 TB_ORDER 테이블에 해당 값이 존재한다는 의미]

PRD_ID	PRD_NAME
P0002	에어컨
P0003	세탁기
P0004	건조기
P0020	수건

답)

```
SELECT PRD_ID , PRD_NAME
  FROM TB_PRD A
 WHERE EXISTS (
                SELECT 'X'
                  FROM TB_ORDER
                 WHERE PRD_ID = A.PRD_ID
               ) ;
```

3. 실습 문제 풀이 (답)

문제2) TB_MEMBER 테이블과 TB_ORDER 테이블을 활용하여 주문을 아직 한번도 하지 않는 회원이면서 GRADE_CD(등급코드) 가 3 인 회원의 MEMBER_ID , MEMBER_NAME , AGE 를 출력해주세요.

[힌트: NOT EXISTS 활용 , 한번도 주문을 안했다는 의미는 TB_ORDER 테이블에 해당 값이 없다는 의미 , 조건에서 AND 조건을 잘 사용해보세요]

MEMBER_ID	MEMBER_NAME	AGE	GRADE_CD
DDDDD	사용자D	30	3
FFFFF	사용자F	35	3

답)

```
SELECT A.MEMBER_ID
     , A.MEMBER_NAME
     , A.AGE
     , A.GRADE_CD
  FROM TB_MEMBER A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_ORDER
                    WHERE A.MEMBER_ID = MEMBER_ID
                  )
   AND GRADE_CD = 3 ;
```

3. 실습 문제 풀이

문제3) [심화]

TB_MEMBER_LIKE 테이블은 회원이 어떤 상품 타입을 선호하는지 정보를 저장한 테이블입니다.
예를 들어 회원 BBBBB 는 선호하는 상품타입이 '가전' 과 '스마트폰' 입니다.
이 때, TB_PRD (상품) 테이블에는 여러가지 상품 타입이 존재합니다.
BBBBB 회원이 선호하는 상품타입 외에 나머지 상품타입들을 아래와 같이 출력해주세요.

PRD_TYPE
주방용품
가전
욕실용품

답)

```
SELECT DISTINCT PRD_TYPE
  FROM TB_PRD A
 WHERE NOT EXISTS (
                   SELECT 1
                     FROM TB_MEMBER_LIKE
                    WHERE A.PRD_TYPE = LIKE_PRD_TYPE
                      AND MEMBER_ID = 'BBBBB'
                   );
```

CASE 문법

현업 사용 정도 : ★ ★ ★ ★ ★

1. CASE 문법 설명
2. CASE 문법 원리 이해하기
3. 실습 문제 풀이

강사 강태우

1. CASE 문법 설명

CASE 문법의 이해를 위해 아래 로직을 생각해봅시다.

TB_MEMBER

MEMBER_ID	MEMBER_NAME	GENDER
AAAAA	사용자A	남
BBBBB	사용자B	여
CCCCC	사용자C	남
DDDDD	사용자D	여
EEEEE	사용자E	남
FFFFF	사용자F	여
GGGGG	사용자G	남
HHHHH	사용자H	(null)
IIIII	사용자I	(null)

GENDER 컬럼의 값이 '남' 이면 'Man' 을 출력하고 '여' 이면 'Woman' 을 출력하고 그 외에는 'notChecked' 를 출력하도록 하는 방법은 뭘까?

1. CASE 문법 설명

CASE 문법의 이해를 위해 아래 로직을 생각해봅시다.

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GENDER
     , CASE WHEN GENDER = '남' THEN 'Man'
            WHEN GENDER = '여' THEN 'Woman'
            ELSE 'notChecked'
             END
              AS 성별영문
  FROM TB_MEMBER ;
```

출력 결과

MEMBER_ID	MEMBER_NAME	GENDER	성별영문
AAAAA	사용자A	남	Man
BBBBB	사용자B	여	Woman
CCCCC	사용자C	남	Man
DDDDD	사용자D	여	Woman
EEEEE	사용자E	남	Man
FFFFF	사용자F	여	Woman
GGGGG	사용자G	남	Man
HHHHH	사용자H	(null)	notChecked
IIIII	사용자I	(null)	notChecked

2. CASE 문법 설명

CASE 문법의 이해를 위해 아래 로직을 생각해봅시다.

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GENDER
     , CASE WHEN GENDER = '남' THEN 'Man'
            WHEN GENDER = '여' THEN 'Woman'
            ELSE 'notChecked'
             END
              AS 성별영문
  FROM TB_MEMBER ;
```

```
CASE   // CASE문법의 시작

    WHEN 조건 THEN 값
    WHEN 조건 THEN 값
     ....
    ELSE 나머지값

END   // CASE 문법의 종료
```

2. CASE 문법 원리 이해하기

CASE 문법이 실행되는 원리를 순서대로 확인해봅시다.

(SELECT에 사용된 경우 출력될 행의 수 만큼 SELECT가 실행 , CASE 문법도 행의 수만큼 실행)

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GENDER
     , CASE WHEN GENDER = '남' THEN 'Man'
            WHEN GENDER = '여' THEN 'Woman'
            ELSE 'notChecked'
             END
              AS 성별영문
  FROM TB_MEMBER ;
```

MEMBER_ID	MEMBER_NAME	GENDER
AAAAA	사용자A	남
BBBBB	사용자B	여
CCCCC	사용자C	남
DDDDD	사용자D	여
EEEEE	사용자E	남
FFFFF	사용자F	여
GGGGG	사용자G	남
HHHHH	사용자H	(null)
IIIII	사용자I	(null)

2. CASE 문법 원리 이해하기

CASE 문법이 실행되는 원리를 순서대로 확인해봅시다.

(SELECT에 사용된 경우 출력될 행의 수 만큼 SELECT가 실행 , CASE 문법도 행의 수만큼 실행)

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GENDER
     , CASE WHEN GENDER = '남' THEN 'Man'
            WHEN GENDER = '여' THEN 'Woman'
            ELSE 'notChecked'
             END
              AS 성별영문
  FROM TB_MEMBER ;
```

MEMBER_ID	MEMBER_NAME	GENDER
AAAAA	사용자A	남
BBBBB	사용자B	여
CCCCC	사용자C	남
DDDDD	사용자D	여
EEEEE	사용자E	남
FFFFF	사용자F	여
GGGGG	사용자G	남
HHHHH	사용자H	(null)
IIIII	사용자I	(null)

2. CASE 문법 원리 이해하기

CASE 문법이 실행되는 원리를 순서대로 확인해봅시다.

(SELECT에 사용된 경우 출력될 행의 수 만큼 SELECT가 실행 , CASE 문법도 행의 수만큼 실행)

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GENDER
     , CASE WHEN GENDER = '남' THEN 'Man'
            WHEN GENDER = '여' THEN 'Woman'
            ELSE 'notChecked'
             END
              AS 성별영문
  FROM TB_MEMBER ;
```

MEMBER_ID	MEMBER_NAME	GENDER
AAAAA	사용자A	남
BBBBB	사용자B	여
CCCCC	사용자C	남
DDDDD	사용자D	여
EEEEE	사용자E	남
FFFFF	사용자F	여
GGGGG	사용자G	남
HHHHH	사용자H	(null)
IIIII	사용자I	(null)

2. CASE 문법 원리 이해하기

주의사항! CASE 문법이 조건이 일치하는 순간 바로 넘어갑니다.

[잘못된 예시]

```
SELECT PRD_NAME
     , PRD_PRICE
     , CASE WHEN PRD_PRICE >= 30000 THEN '3만원이상'
            WHEN PRD_PRICE >= 100000 THEN '10만원이상'
            WHEN PRD_PRICE >= 1000000 THEN '100만원이상'
            ELSE '1000만원이상' END AS 얼마이상인가요
  FROM TB_PRD ;
```

에어컨 가격도 30000원 이상이니까
첫째 조건에서 일치하므로
바로 출력이 되어버리는 문제가 발생

PRD_NAME	PRD_PRICE	얼마이상인가요
헤어드라이기	30000	3만원이상
에어컨	1500000	3만원이상
세탁기	600000	3만원이상
건조기	800000	3만원이상
노트북	1500000	3만원이상
데스크탑	2000000	3만원이상
태블릿	800000	3만원이상
애플14	1200000	3만원이상
갤럭시s23	1500000	3만원이상
조아샴푸	20000	1000만원이상
주전자	20000	1000만원이상
전기밥솥	80000	3만원이상
냄비	30000	3만원이상
칼	15000	1000만원이상
수세미	5000	1000만원이상
곰팡이제거제	10000	1000만원이상
샤워기	50000	3만원이상
린스	20000	1000만원이상
수건	5000	1000만원이상

2. CASE 문법 원리 이해하기

주의사항! CASE 문법이 조건이 일치하는 순간 바로 넘어갑니다.

[올바른 예시]

```
SELECT PRD_NAME
     , PRD_PRICE
     , CASE WHEN PRD_PRICE >= 1000000 THEN '100만원이상'
            WHEN PRD_PRICE >= 100000 THEN '10만원이상'
            WHEN PRD_PRICE >= 30000 THEN '3만원이상'
            WHEN PRD_PRICE <  30000 THEN '3만원미만'
            ELSE '1000만원이상' END AS 얼마이상인가요
  FROM TB_PRD ;
```

TIP) 범위 조건으로 CASE를 쓴다면
크기를 고려하여 실행해야 합니다.

PRD_NAME	PRD_PRICE	얼마이상인가요
헤어드라이기	30000	3만원이상
에어컨	1500000	100만원이상
세탁기	600000	10만원이상
건조기	800000	10만원이상
노트북	1500000	100만원이상
데스크탑	2000000	100만원이상
태블릿	800000	10만원이상
애플14	1200000	100만원이상
갤럭시s23	1500000	100만원이상
조아샴푸	20000	3만원미만
주전자	20000	3만원미만
전기밥솥	80000	3만원이상
냄비	30000	3만원이상
칼	15000	3만원미만
수세미	5000	3만원미만
곰팡이제거제	10000	3만원미만
샤워기	50000	3만원이상
린스	20000	3만원미만
수건	5000	3만원미만

3. 실습 문제 풀이

문제1) CASE 문법을 이용하여 각 회원의 등급코드(GRADE_CD) 별로 등급이름을 출력하려고 합니다.

출력하려는 값은 다음과 같습니다.

[1 -'브론즈' , 2-'실버' , 3-'골드' , 4-'VIP' , 5-'VVIP' , 그외-'X']

위 등급이름을 참조하여 아래와 같이 데이터를 출력해주세요. (TB_MEMBER 테이블 활용)

MEMBER_ID	MEMBER_NAME	GRADE_CD	등급이름
AAAAA	사용자A	1	브론즈
BBBBB	사용자B	2	실버
CCCCC	사용자C	1	브론즈
DDDDD	사용자D	3	골드
EEEEE	사용자E	1	브론즈
FFFFF	사용자F	3	골드
GGGGG	사용자G	2	실버
HHHHH	사용자H	5	VVIP
IIIII	사용자I	4	VIP

3. 실습 문제 풀이

문제2) 문제1번에서 출력한 내용을 오랜만에 '조인'을 이용해서도 풀어봅시다.

힌트 : TB_MEMBER 테이블과 TB_GRADE 테이블을 GRADE_CD 컬럼으로 조인하여 원하는 컬럼만 가져온다.

이너조인으로 풀어주세요. (여기서는 사용자가 모두 GRADE_CD 가 존재한다고 가정합니다)

MEMBER_ID	MEMBER_NAME	GRADE_CD	등급이름
AAAAA	사용자A	1	브론즈
BBBBB	사용자B	2	실버
CCCCC	사용자C	1	브론즈
DDDDD	사용자D	3	골드
EEEEE	사용자E	1	브론즈
FFFFF	사용자F	3	골드
GGGGG	사용자G	2	실버
HHHHH	사용자H	5	VVIP
IIIII	사용자I	4	VIP

3. 실습 문제 풀이

문제3) TB_MEMBER 테이블의 GRADE_CD 값을 기준으로 등급이 4이상이면 추가 할인 쿠폰(10%)를 발행하기로 했습니다. 아래와 같이 대상/비대상을 출력할 때 CASE 문법을 이용해주세요.

MEMBER_ID	MEMBER_NAME	GRADE_CD	쿠폰대상여부
AAAAA	사용자A	1	비대상
BBBBB	사용자B	2	비대상
CCCCC	사용자C	1	비대상
DDDDD	사용자D	3	비대상
EEEEE	사용자E	1	비대상
FFFFF	사용자F	3	비대상
GGGGG	사용자G	2	비대상
HHHHH	사용자H	5	대상
IIIII	사용자I	4	대상

3. 실습 문제 풀이

문제1) CASE WHEN 절을 각 회원의 등급코드(GRADE_CD) 별로 등급이름을 출력하려고 합니다.

출력하려는 값은 다음과 같습니다.

[1 -'브론즈' , 2-'실버' , 3-'골드' , 4-'VIP' , 5-'VVIP' , 그외-'X']

위 등급이름을 참조하여 아래와 같이 데이터를 출력해주세요. (TB_MEMBER 테이블 활용)

답)

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GRADE_CD
     , CASE WHEN GRADE_CD = 1 THEN '브론즈'
            WHEN GRADE_CD = 2 THEN '실버'
            WHEN GRADE_CD = 3 THEN '골드'
            WHEN GRADE_CD = 4 THEN 'VIP'
            WHEN GRADE_CD = 5 THEN 'VVIP'
            ELSE 'X'
             END AS 등급이름
  FROM TB_MEMBER ;
```

MEMBER_ID	MEMBER_NAME	GRADE_CD	등급이름
AAAAA	사용자A	1	브론즈
BBBBB	사용자B	2	실버
CCCCC	사용자C	1	브론즈
DDDDD	사용자D	3	골드
EEEEE	사용자E	1	브론즈
FFFFF	사용자F	3	골드
GGGGG	사용자G	2	실버
HHHHH	사용자H	5	VVIP
IIIII	사용자I	4	VIP

3. 실습 문제 풀이

문제2) 문제1번에서 출력한 내용을 오랜만에 '조인'을 이용해서도 풀어봅시다.

힌트 : TB_MEMBER 테이블과 TB_GRADE 테이블을 GRADE_CD 컬럼으로 조인하여 원하는 컬럼만 가져온다.

이너조인으로 풀어주세요. (여기서는 사용자가 모두 GRADE_CD 가 존재한다고 가정합니다)

답)

```
SELECT A.MEMBER_ID
     , A.MEMBER_NAME
     , A.GRADE_CD
     , B.GRADE_NAME
  FROM TB_MEMBER A
     , TB_GRADE B
 WHERE A.GRADE_CD = B.GRADE_CD ;
```

MEMBER_ID	MEMBER_NAME	GRADE_CD	등급이름
AAAAA	사용자A	1	브론즈
BBBBB	사용자B	2	실버
CCCCC	사용자C	1	브론즈
DDDDD	사용자D	3	골드
EEEEE	사용자E	1	브론즈
FFFFF	사용자F	3	골드
GGGGG	사용자G	2	실버
HHHHH	사용자H	5	VVIP
IIIII	사용자I	4	VIP

3. 실습 문제 풀이

문제3) TB_MEMBER 테이블의 GRADE_CD 값을 기준으로 등급이 4이상이면 추가 할인 쿠폰(10%)를 발행하기로 했습니다. 아래와 같이 대상/비대상을 출력할 때 CASE 문법을 이용해주세요.

답)

```
SELECT MEMBER_ID
     , MEMBER_NAME
     , GRADE_CD
     , CASE WHEN GRADE_CD >= 4 THEN '대상'
            ELSE '비대상'
             END AS 쿠폰대상여부
  FROM TB_MEMBER ;
```

MEMBER_ID	MEMBER_NAME	GRADE_CD	쿠폰대상여부
AAAAA	사용자A	1	비대상
BBBBB	사용자B	2	비대상
CCCCC	사용자C	1	비대상
DDDDD	사용자D	3	비대상
EEEEE	사용자E	1	비대상
FFFFF	사용자F	3	비대상
GGGGG	사용자G	2	비대상
HHHHH	사용자H	5	대상
IIIII	사용자I	4	대상

MERGE 문법

현업 사용 정도 : ★ ★ ★ ★ ☆

1. MERGE 문법 설명
2. MERGE 사용 이유
3. 실습 문제 풀이

강사 강태우

1. MERGE 문법 설명

MERGE문법 의 이해를 위해 아래 로직을 생각해봅시다.

TB_MEMBER_TEL (회원연락처) 테이블에 데이터를 등록할 때

특정 회원이 이미 해당 연락처구분코드로 값을 가지고 있는지 확인을 해야 합니다.
이미 존재한다면 UPDATE를 하면 되고 그렇지 않다면 INSERT 를 진행하면 됩니다.

예를 들어 다음과 같은 데이터가 입력되었습니다.
[MEMBER_ID : 'BBBBB' , TEL_DV_CD : '휴대폰' , TEL_NO : '010-7777-7777']
[MEMBER_ID = 'BBBBB' , 'TEL_DV_CD : '회사' , TEL_NO : '02-5678-1234']

우리는 UPDATE / INSERT 중에 어떤 걸 선택해야 할까요?

1. MERGE 문법 설명

MERGE문법 의 이해를 위해 아래 로직을 생각해봅시다.

기본적인 방법은

1. 먼저 SELECT 로 해당 조건의 데이터가 있는지 찾고
2. 있으면 UPDATE , 없으면 INSERT 한다 입니다.

```
SELECT '1'
  FROM TB_MEMBER_TEL
 WHERE MEMBER_ID = 'BBBBB'
   AND TEL_DV_CD = '휴대폰' ;
```

SQL | 인출된 모든 행: 1(0.001초)

'1'
1

조건에 일치하는 데이터가 있음

```
UPDATE TB_MEMBER_TEL
   SET TEL_NO = '010-7777-7777'
 WHERE MEMBER_ID = 'BBBBB'
   AND TEL_DV_CD = '휴대폰' ;
```

1. MERGE 문법 설명

MERGE문법 의 이해를 위해 아래 로직을 생각해봅시다.

기본적인 방법은

1. 먼저 SELECT 로 해당 조건의 데이터가 있는지 찾고
2. 있으면 UPDATE , 없으면 INSERT 한다 입니다.

```
SELECT '1'
  FROM TB_MEMBER_TEL
 WHERE MEMBER_ID = 'BBBBB'
   AND TEL_DV_CD = '회사' ;
```

SQL | 인출된 모든 행: 0(0.002초)

'1'

조건에 일치하는 데이터가 없음

```
INSERT INTO TB_MEMBER_TEL (
        MEMBER_ID
      , TEL_DV_CD
      , TEL_NO
) VALUES (
        'BBBBB'
      , '회사'
      , '02-5678-1234'
) ;
```

2. MERGE 사용 이유

MERGE문법은 위의 과정을 줄여 성능을 향상시켜줄 수 있습니다.

즉, SELECT 로 판단을 해서 UPDATE 나 INSERT 하는 행위를 한번에 처리를 해줍니다.

```
MERGE INTO TB_MEMBER_TEL A
USING DUAL
   ON ( A.MEMBER_ID = 'BBBBB' AND TEL_DV_CD = '휴대폰' )
WHEN MATCHED THEN

UPDATE
   SET TEL_NO = '010-7777-7777'
WHEN NOT MATCHED THEN

INSERT ( MEMBER_ID , TEL_DV_CD , TEL_NO )
VALUES ( 'BBBBB'    , '휴대폰' , '010-7777-7777' ) ;
```

MERGE INTO 대상테이블
USING 비교할테이블 (없으면 DUAL)
ON (찾아볼 조건)

위 조건에 해당하는 값이 있다면

UPDATE 로 진행

위 조건에 해당하는 값이 없다면

INSERT 로 진행

2. MERGE 사용 이유

문제) 아래 쿼리에서 값만 적절히 변경을 해서 아래 데이터에 대해 MERGE 문을 처리해주세요.

[MEMBER_ID = 'BBBBB' , 'TEL_DV_CD : '회사' , TEL_NO : '02-5678-1234']

```
MERGE INTO TB_MEMBER_TEL A
USING DUAL
   ON ( A.MEMBER_ID = 'BBBBB' AND TEL_DV_CD = '휴대폰' )

WHEN MATCHED THEN

UPDATE
   SET TEL_NO = '010-7777-7777'

WHEN NOT MATCHED THEN

INSERT ( MEMBER_ID , TEL_DV_CD , TEL_NO )
VALUES ( 'BBBBB'    , '휴대폰' , '010-7777-7777' ) ;
```

MERGE의 또 다른 예시를 보기위해 아래 데이터를 복사&실행해주세요.

```
DROP TABLE 직원;
DROP TABLE 직원_신입 ;

CREATE TABLE 직원 (
    직원ID VARCHAR2(30) PRIMARY KEY ,
    직원이름 VARCHAR2(50) NOT NULL ,
    연봉    NUMBER NOT NULL
) ;

INSERT INTO 직원 VALUES ( 'A0001' , '김현명' , 4000) ;
INSERT INTO 직원 VALUES ( 'A0002' , '강태진' , 5000) ;
INSERT INTO 직원 VALUES ( 'A0003' , '손지창' , 5000) ;

CREATE TABLE 직원_신입 (
    직원ID VARCHAR2(30) PRIMARY KEY ,
    직원이름 VARCHAR2(50) NOT NULL ,
    연봉    NUMBER NOT NULL
) ;

INSERT INTO 직원_신입 VALUES ( 'A0001' , '김현명' , 4000) ;
INSERT INTO 직원_신입 VALUES ( 'A0002' , '강태진' , 5000) ;
INSERT INTO 직원_신입 VALUES ( 'A0003' , '손지창' , 6000) ;
INSERT INTO 직원_신입 VALUES ( 'A0004' , '신입원' , 3400) ;
INSERT INTO 직원_신입 VALUES ( 'A0005' , '신입투' , 3400) ;

COMMIT ;
```

직원 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	5000

직원_신입 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	6000
A0004	신입원	3400
A0005	신입투	3400

직원 테이블은 직원의 정보를 저장하는 테이블입니다.
직원_신입 테이블은 새로 들어온 신입들의 정보를 가지고 있는 테이블입니다.

잘 보면 두 테이블은 동일한 직원ID 를 가지고 있거나 ,
직원_신입 테이블만 가지고 있는 직원ID 를 가지고 있습니다.

직원_신입 테이블의 데이터를 직원 테이블에 병합(MERGE)을
하여 직원의 데이터를 최신으로 유지하려고 합니다.
직원ID 컬럼을 기준으로 기존에 존재하는 데이터는 UPDATE ,
새로운 데이터는 INSERT 가 되도록 해주세요.

직원 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	5000

직원_신입 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	6000
A0004	신입원	3400
A0005	신입투	3400

아래 문법을 천천히 작성하며 의미를 이해해보고 실행 후 직원 테이블을 조회해보세요.

```
MERGE INTO 직원 A  -- 데이터를 병합하려는 대상 테이블
USING 직원_신입 B   -- 병합에 사용될 테이블
   ON ( A.직원ID = B.직원ID) --비교조건
                             --두 값이 똑같으면 UPDATE
                             --그렇지 않으면 INSERT

WHEN MATCHED THEN  -- 두 값이 똑같을 경우 (여기서는 A0001 , A0002 , A0003 )
UPDATE
   SET A.직원이름 = B.직원이름
     , A.연봉     = B.연봉

WHEN NOT MATCHED THEN -- 두 값이 다를 경우 (여기서는 A0004 , A0005 )
INSERT ( A.직원ID , A.직원이름 , A.연봉 )
VALUES ( B.직원ID , B.직원이름 , B.연봉 ) ;
```

직원 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	5000

직원_신입 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	6000
A0004	신입원	3400
A0005	신입투	3400

병합 후 직원 테이블

직원ID	직원이름	연봉
A0001	김현명	4000
A0002	강태진	5000
A0003	손지창	6000
A0005	신입투	3400
A0004	신입원	3400

3. 실습 문제 풀이

문제) 회원ID 가 ‘CCCCC’ 인 회원이 연락처 정보를 입력했습니다.

아래와 같이 데이터가 입력되었는데 MERGE 문을 이용해서 적절하게 처리를 해주세요.

[MEMBER_ID = ‘CCCCC’ , ‘TEL_DV_CD : ‘휴대폰’ , TEL_NO : ’010-8888-8888’]

- 힌트 : TB_MEMBER_TEL 테이블에서는 MEMBER_ID 와 TEL_DV_CD 조합으로 유일한 값을 식별하는 Primary Key 입니다.
  즉, 조건을 줄 때 MEMBER_ID 와 TEL_DV_CD 를 이용하면 해당하는 조건의 값이 있는지 없는지 유일하게 판별이 가능합니다.

3. 실습 문제 풀이

답)

```
MERGE INTO TB_MEMBER_TEL
USING DUAL
    ON ( MEMBER_ID = 'CCCCC' AND TEL_DV_CD = '휴대폰' )

WHEN MATCHED THEN

UPDATE
    SET TEL_NO = '010-8888-8888'

WHEN NOT MATCHED THEN

INSERT ( MEMBER_ID , TEL_DV_CD , TEL_NO )
VALUES ( 'CCCCC'    , '휴대폰' , '010-8888-8888' ) ;
```

EXISTS
CASE 문법
MERGE
END