

## 7. 테이블 구조를 결정하는 DDL로 테이블 생성, 변경, 삭제

CREATE TABLE로 테이블 구조 정의하는 방법을 학습한다.

ALTER TABLE문은 테이블의 구조를 변경한다.

DROP TABLE문은 기존 테이블을 제거한다.

테이블명 변경하는 RENAME문, 테이블의 모든 로우를 제거해 TRUNCATE 문을 학습한다.

※ DDL : 테이블의 구조 자체를 생성, 수정, 제거하도록 하는 명령문 집합이다.

데이터를 담고 있는 객체가 테이블이다. 테이블은 DBMS상에서 가장 기본적인 객체로 컬럼과 레코드로 구성된 2차원 표의 객체이다. SQL을 이용해 데이터를 조회, 삭제, 입력 수정할 대상이며 그 결과를 담고 있는 것이 바로 테이블이다.

1) CREATE TABLE 테이블 구조 정의: 새로운 테이블을 생성하기 위한 명령어로 CREATE가 있다.

| CREATE TABLE table_name<br>(column_name data_type expr, ...); |   |   |   |   |   |   |   |   |   |   |   |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|--|--|--|--|--|
| 칼럼을 정의 할 때 지정할 수 있는 자료형                                       |   |   |   |   |   |   |   |   |   |   |   |  |  |  |  |  |
| 이름  |   |   |   |   |   |   |   |   |   |   |   |  |  |  |  |  |
| CHAR(size)  | <p>고정 길이 문자 데이터. 입력된 자료의 길이와는 상관없이 정해진 길이만큼 저장 영역 차지(문자길이+공백). 데이터를 입력하지 않으면 NULL이 자동으로 입력되고, 지정된 길이보다 긴 데이터가 입력되면 오류가 발생.</p> <p>최대크기: 2,000 byte / 최소크기: 1 바이트</p> <p>CHAR(10) : 고정 형식의 10자리 문자열</p> <p>입력 <table><tr><td>H</td><td>e</td><td>l</td><td>l</td><td>o</td></tr></table> (5자리)</p> <p>↓</p> <p>저장 <table><tr><td>H</td><td>e</td><td>l</td><td>l</td><td>o</td><td></td><td></td><td></td><td></td><td></td></tr></table> (10자리)</p> | H | e | l | l | o | H | e | l | l | o |  |  |  |  |  |
| H   | e   | l | l | o |   |   |   |   |   |   |   |  |  |  |  |  |
| H   | e   | l | l | o |   |   |   |   |   |   |   |  |  |  |  |  |
| VARCHAR2(size)  | <p>가변 길이 문자 데이터. 실제 입력된 문자열의 길이만큼 저장 영역을 차지.</p> <p>최대크기: 4,000 byte / 최소크기: 1 바이트</p> <p>VARCHAR2(10) : 가변 형식의 10자리 문자열</p> <p>입력 <table><tr><td>H</td><td>e</td><td>l</td><td>l</td><td>o</td></tr></table> (5자리)</p> <p>↓</p> <p>저장 <table><tr><td>H</td><td>e</td><td>l</td><td>l</td><td>o</td></tr></table> (5자리)</p>   | H | e | l | l | o | H | e | l | l | o |  |  |  |  |  |
| H   | e   | l | l | o |   |   |   |   |   |   |   |  |  |  |  |  |
| H   | e   | l | l | o |   |   |   |   |   |   |   |  |  |  |  |  |
| NUMBER<br>NUMBER(p)<br>NUMBER(p, s)                           | <p>형식: <b>NUMBER(전체 자릿수(p), 소수점이하 자릿수(s))</b>로 표현되는 숫자 데이터 타입</p> <p>전체 자릿수만 지정하면 소수점 이하는 반올림되어 정수값만 저장된다. 전체 자릿수도 소수점 이하 자릿수도 모두 생략하면 입력한 데이터값만큼 공간이 할당된다. 둘다 지정한 경우에는 실수 형태의 값이 저장된다. 최대 22 바이트이다.</p> <p>정밀도(p) : 1 ~ 38. 디폴트 값은 38 / 스케일(s) : -84 ~ 127. 디폴트 값은 0</p> <p>NUMBER(3): 3자리 수치 (999까지 표현이 가능)</p> <p>NUMBER(6, 2): 소수점 2자리를 포함한 6자리 수치</p>  |   |   |   |   |   |   |   |   |   |   |  |  |  |  |  |

단위는 byte나 문자수가 된다. char(6 byte)와 char(6 char)으로 정의가 가능한데 그때 영문일 경우는 둘 다 6개의 데이터를 저장할 수 있으며 한글은 char(6 char)에만 6개의 글자를 저장할 수 있다.

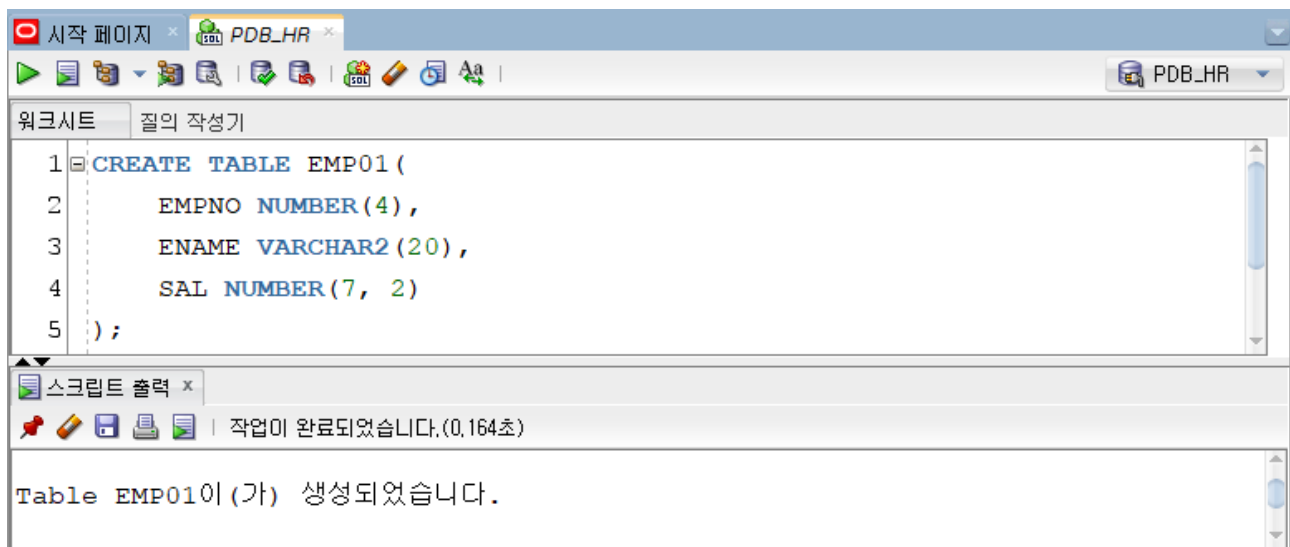
|           |   |
|-----------|---|
| BLOB      | 대용량의 바이너리 데이터를 저장하기 위한 데이터 타입.<br>최대크기: 4GB                                 |
| CLOB      | 대용량의 텍스트 데이터를 저장하기 위한 데이터 타입.<br>최대크기: 4GB                                  |
| DATE      | 날짜 형식을 저장하기 위한 데이터 타입.  |
| TIMESTAMP | DATE 데이터 타입의 확장된 형태.  |
| ROWID     | 테이블 내 행의 고유 주소를 가지는 64진수 문자 타입.<br>해당 6 바이트(제한된 ROWID) 또는 10 바이트(확장된 ROWID) |

#### • 테이블명과 컬럼명을 부여하기 위한 규칙

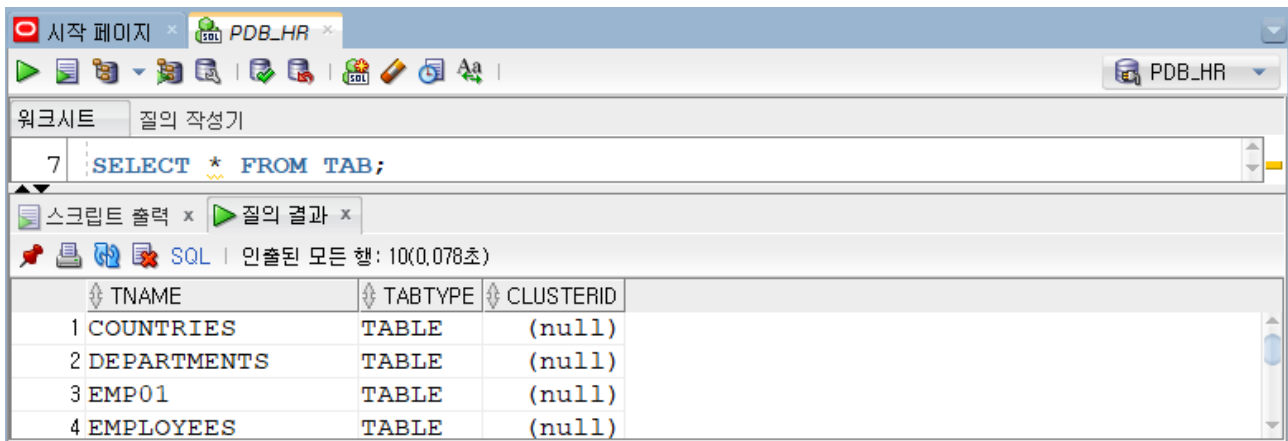
- 반드시 문자(A-Z, a-z)로 시작해야 하며, 컬럼명은 최대 30바이트까지 가능하다.
- A~Z까지의 대·소문자와 0~9까지의 숫자, 특수기호는 (, \$, #)만 가능하다.
- 오라클에서 사용되는 예약어나 다른 객체명과 중복하여 생성할 수 없다.
- 공백 허용하지 않는다.
- 대소문자 구별이 없다. 소문자로 저장하려면 ''로 묶어 주어야 한다.
- 한 테이블에 사용 가능한 컬럼은 최대 255개까지다.

사원번호, 사원명, 급여 3개의 컬럼으로 구성된 EMP01 테이블

```
CREATE TABLE EMP01(
    EMPNO NUMBER(4),
    ENAME VARCHAR2(20),
    SAL NUMBER(7, 2)
);
```



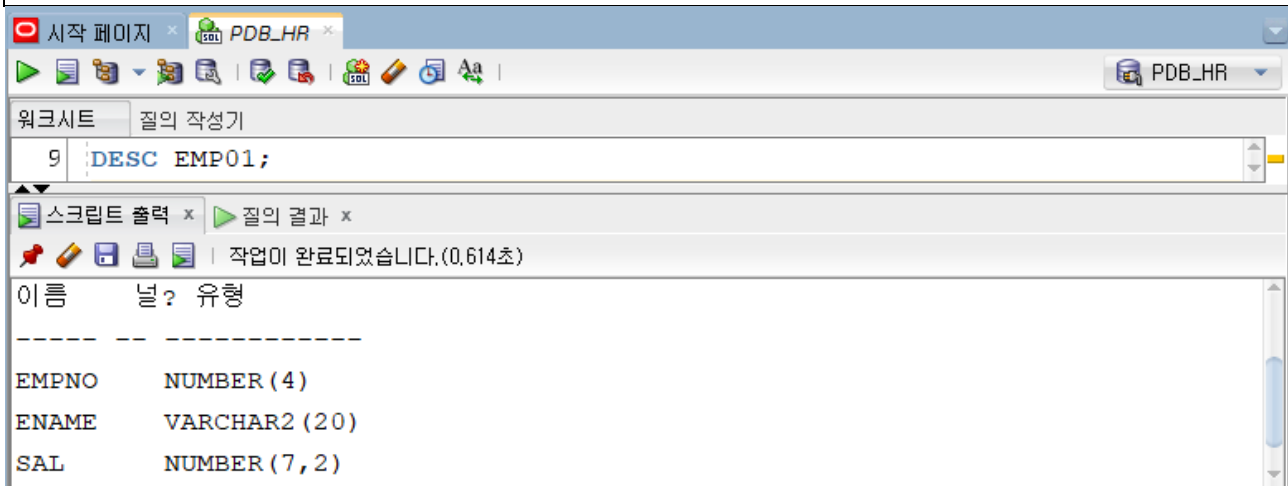
```
SELECT * FROM TAB;
```



```
SELECT * FROM EMP01;
```

테이블의 구조 확인.

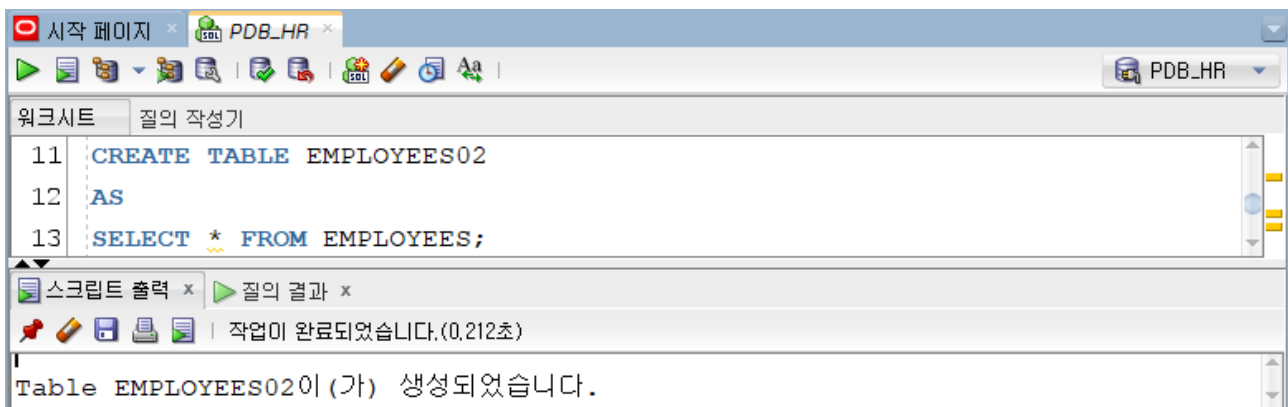
```
DESC EMP01;
```



## 2) 기존 테이블 복사

기존 테이블과 동일한 구조와 내용을 갖는 테이블을 생성한다.

```
CREATE TABLE EMPLOYEES02
AS
SELECT * FROM EMPLOYEES;
```



SELECT문을 위와 같이 AS절 다음에 기술하여 출력방향이 화면이 아닌 새로운 테이블로 하면 테이블이

생성된다. 즉, EMPLOYEES02는 EMPLOYEES 테이블이 그대로 복사되어 내용과 구조가 동일한 테이블이 생성된다.

```
SELECT * FROM TAB;
```

| TNAME         | TABTYPE | CLUSTERID |
|---------------|---------|-----------|
| 1 COUNTRIES   | TABLE   | (null)    |
| 2 DEPARTMENTS | TABLE   | (null)    |
| 3 EMP01       | TABLE   | (null)    |
| 4 EMPLOYEES   | TABLE   | (null)    |
| 5 EMPLOYEES02 | TABLE   | (null)    |

```
SELECT * FROM EMPLOYEES02;
```

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME   | EMAIL     | PHONE_NUMBER | HIRE_DATE    | J           |
|-------------|------------|-------------|-----------|--------------|--------------|-------------|
| 1           | 100        | Steven      | King      | SKING        | 515.123.4567 | 03/06/17 AD |
| 2           | 101        | Neena       | Kochhar   | NKOCHHAR     | 515.123.4568 | 05/09/21 AD |
| 3           | 102        | Lex         | De Haan   | LDEHAAN      | 515.123.4569 | 01/01/13 AD |
| 4           | 103        | Alexander   | Hunold    | AHUNOLD      | 590.423.4567 | 06/01/03 IT |
| 5           | 104        | Bruce       | Ernst     | BERNST       | 590.423.4568 | 07/05/21 IT |
| 6           | 105        | David       | Austin    | DAUSTIN      | 590.423.4569 | 05/06/25 IT |
| 7           | 106        | Valli       | Pataballa | VPATABAL     | 590.423.4560 | 06/02/05 IT |
| 8           | 107        | Diana       | Lorentz   | DLORENTZ     | 590.423.5567 | 07/02/07 IT |
| 9           | 108        | Nancy       | Greenberg | NGREENBE     | 515.124.4569 | 02/08/17 FI |
| 10          | 109        | Daniel      | Faviet    | DFAVIET      | 515.124.4169 | 02/08/16 FI |
| 11          | 110        | John        | Chen      | JCHEN        | 515.124.4269 | 05/09/28 FI |
| 12          | 111        | Ismael      | Sciarra   | ISCIARRA     | 515.124.4369 | 05/09/30 FI |
| 13          | 112        | Jose Manuel | Urman     | JMURMAN      | 515.124.4469 | 06/03/07 FI |

### 3) ALTER TABLE로 테이블 구조 변경

ALTER TABLE 명령어는 테이블에서 칼럼의 추가, 삭제, 칼럼의 타입이나 길이를 변경할 때 사용한다.

- ADD(컬럼명 자료형)절을 사용하여 새로운 칼럼을 추가한다.
- MODIFY(컬럼명 자료형)절을 사용하여 기존 칼럼을 수정한다.
- DROP COLUMN 컬럼명절을 사용하여 기존 칼럼을 삭제한다.

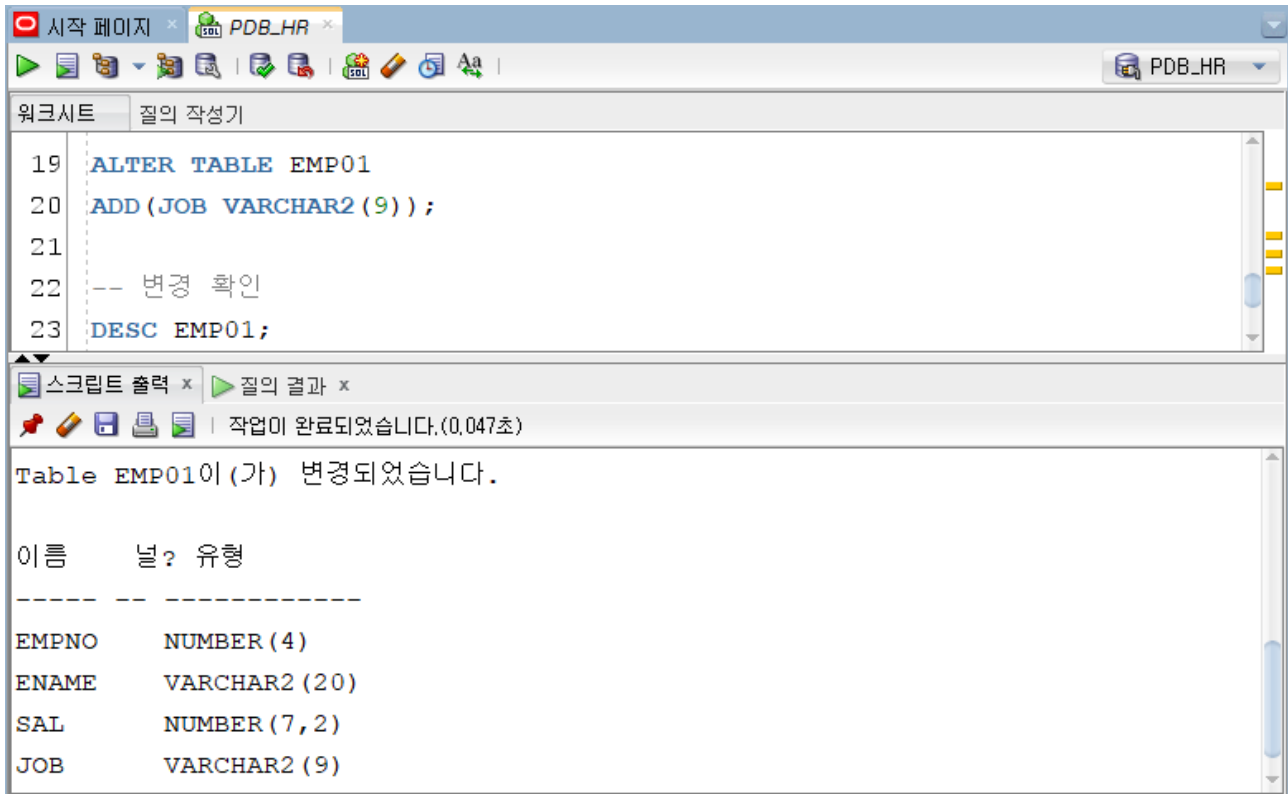
#### ① ALTER TABLE로 새로운 칼럼 추가

```
ALTER TABLE table_name
ADD(column_name data_type expr, ...);
```

<예> EMP01 테이블에 문자 타입의 직급(JOB) 칼럼을 추가

```
ALTER TABLE EMP01
ADD(JOB VARCHAR2(9));

-- 변경 확인
DESC EMP01;
```



<문제> 이미 존재하는 EMP01 테이블에 입사일 칼럼(CREDATE)을 날짜형으로 추가하라.

② ALTER TABLE로 기존 칼럼 수정

ALTER TABLE명령어에 MODIFY절을 사용하여 칼럼을 수정한다. 데이터의 타입, 크기를 변경할 수 있다.

```
ALTER TABLE table_name  
MODIFY(column_name data_type expr, ...);
```

- 해당 칼럼에 자료가 없는 경우

칼럼의 데이터 타입을 변경할 수 있다.

칼럼의 크기를 변경할 수 있다.

- 해당 칼럼에 자료가 있는 경우

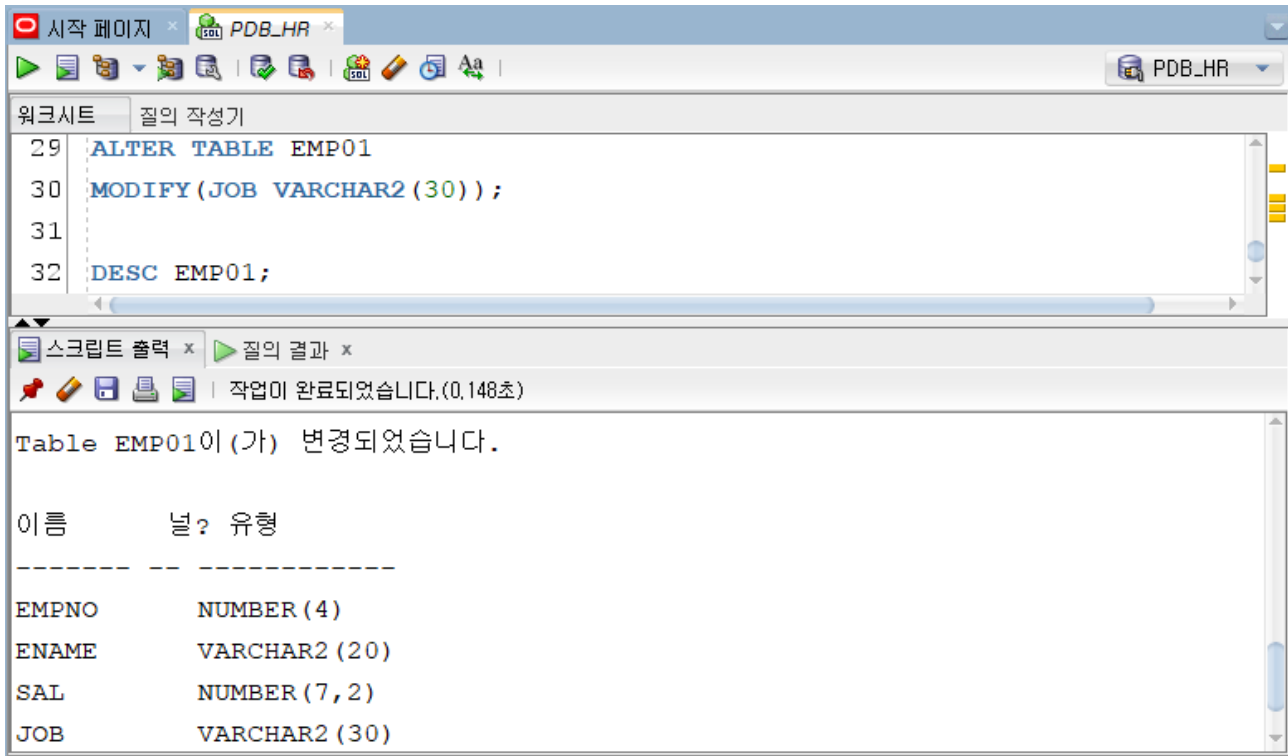
칼럼의 데이터 타입을 변경할 수 없다.

크기를 늘릴 수는 있지만 현재 가지고 있는 데이터 크기보다 작은 크기로 변경할 수 없다.

<예> 직급을 최대 30자까지 입력할 수 있도록 크기 수정

```
ALTER TABLE EMP01
MODIFY(JOB VARCHAR2(30));

DESC EMP01;
```



### ③ ALTER TABLE로 기존 칼럼명 변경

```
ALTER TABLE table_name
RENAME COLUMN old_name TO new_name;
```

<예> 입사일 컬럼의 이름을 CDATE에서 REGDATE로 컬럼명을 변경.

```
ALTER TABLE EMP01
RENAME COLUMN CDATE TO REGDATE;

DESC EMP01;
```

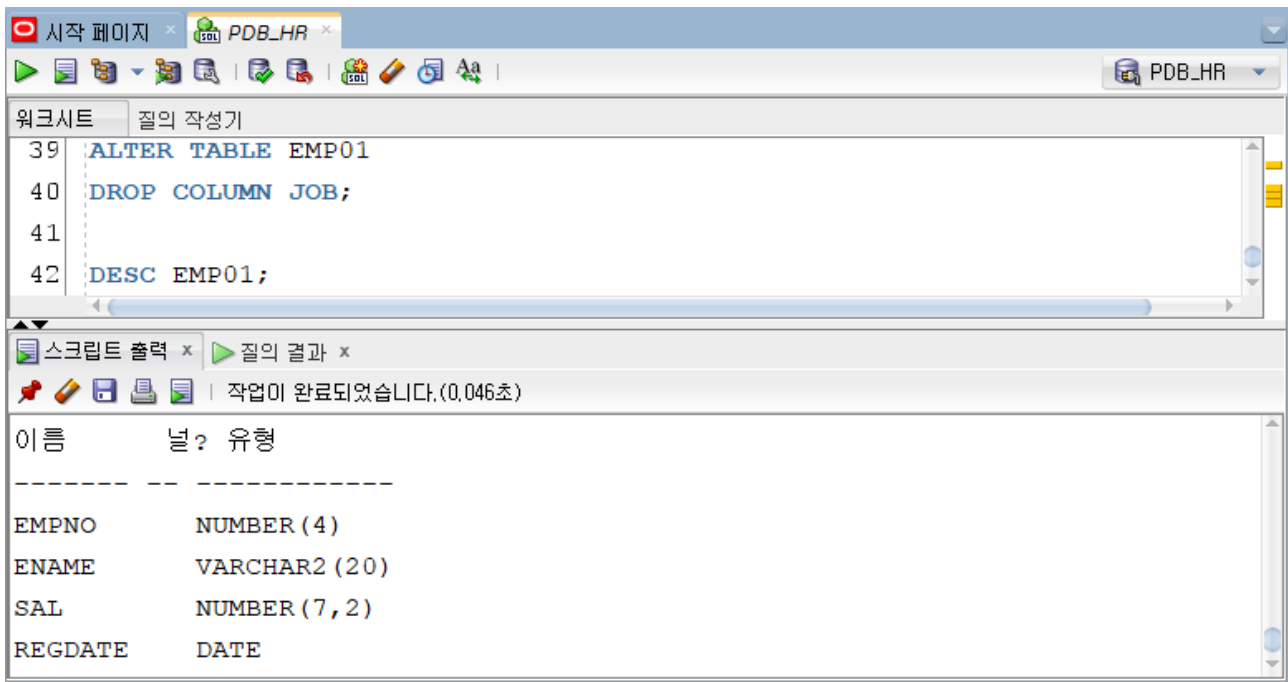
### ④ ALTER TABLE로 기존 칼럼 삭제

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

<예> 직급(JOB) 칼럼을 삭제

```
ALTER TABLE EMP01
DROP COLUMN JOB;

DESC EMP01;
```



#### 4) DROP TABLE로 테이블 구조 삭제

DROP TABLE문은 기존 테이블을 삭제한다. 테이블을 제거하면 테이블에 저장되어 있는 데이터도 함께 제거된다. 제거된 데이터는 다시 복구하기 힘들기 때문에 명령어 사용시 유의한다.

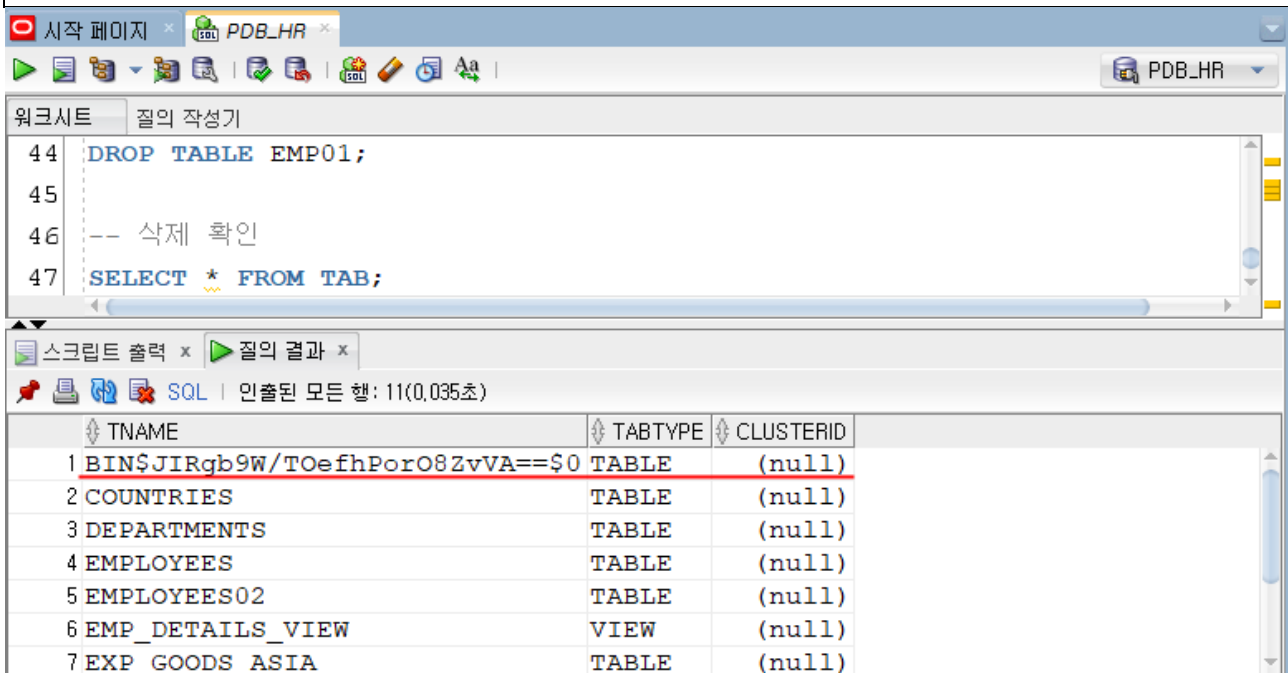
**DROP TABLE table\_name;**

<예> EMP01 테이블을 삭제

```

DROP TABLE EMP01;
-- 삭제 확인
SELECT * FROM TAB;

```

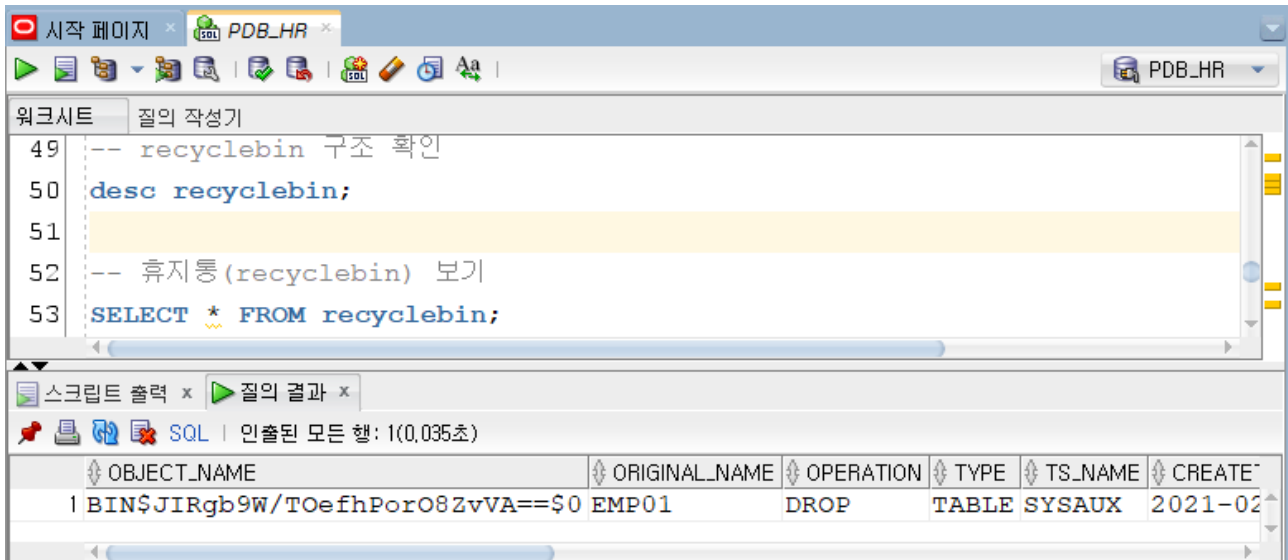


-- recyclebin 구조 확인

```
desc recyclebin;
```

```
-- 휴지통(recyclebin) 보기
```

```
SELECT * FROM recyclebin;
```



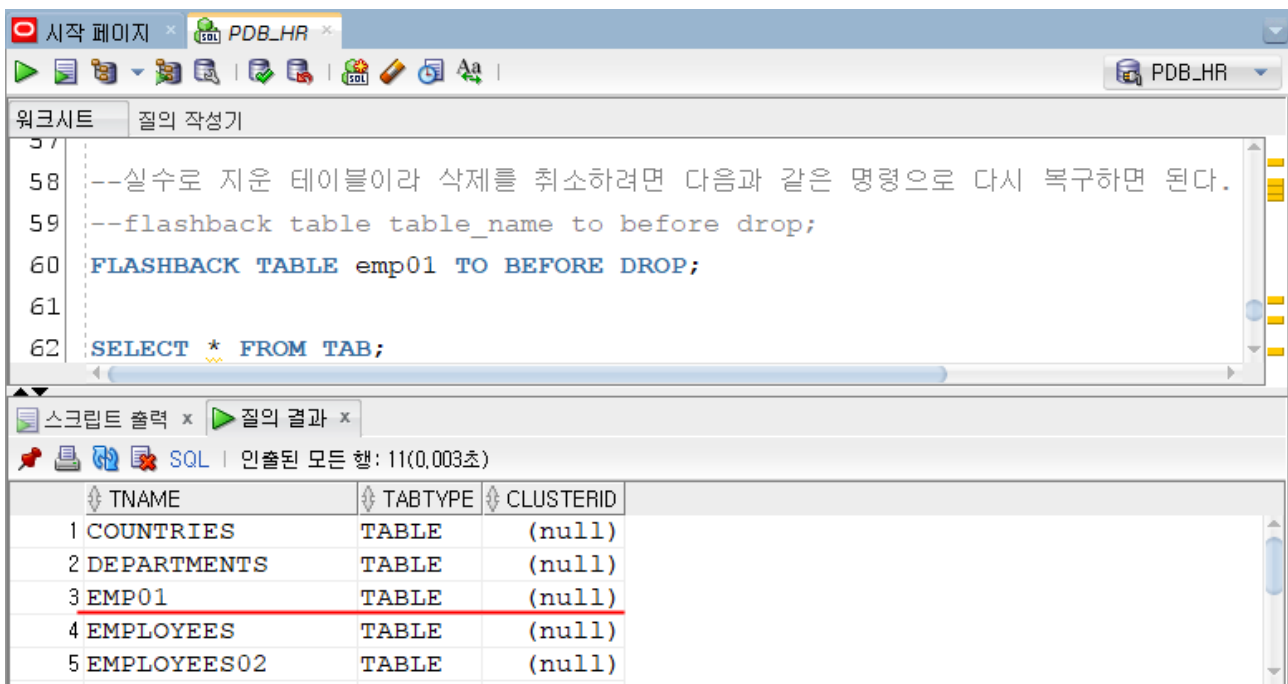
```
-- 휴지통 비우기
```

```
purge recyclebin;
```

```
--실수로 지운 테이블이라 삭제를 취소하려면 다음과 같은 명령으로 다시 복구하면 된다.
```

```
--flashback table table_name to before drop;
```

```
FLASHBACK TABLE emp01 TO BEFORE DROP;
```





-- 새로운 이름으로 복원하는 방법

```
FLASHBACK TABLE emp01 TO BEFORE DROP  
RENAME TO emp02;
```

-- 휴지통에 넣지 않고 바로 테이블을 삭제하려면 다음과 같은 명령으로 휴지통에 넣지 않고 삭제를 할 수 있다.

```
--drop table table_name purge;  
drop table emp01 purge;
```

#### \* 테이블 삭제와 무결성 제약 조건

삭제하고자 하는 테이블의 기본 키나 고유 키를 다른 테이블에서 참조해서 사용하는 경우에는 해당 테이블을 제거할 수 없다. 이러한 경우에는 참조하는 테이블을 먼저 제거한 후에 해당 테이블을 삭제해야 한다.

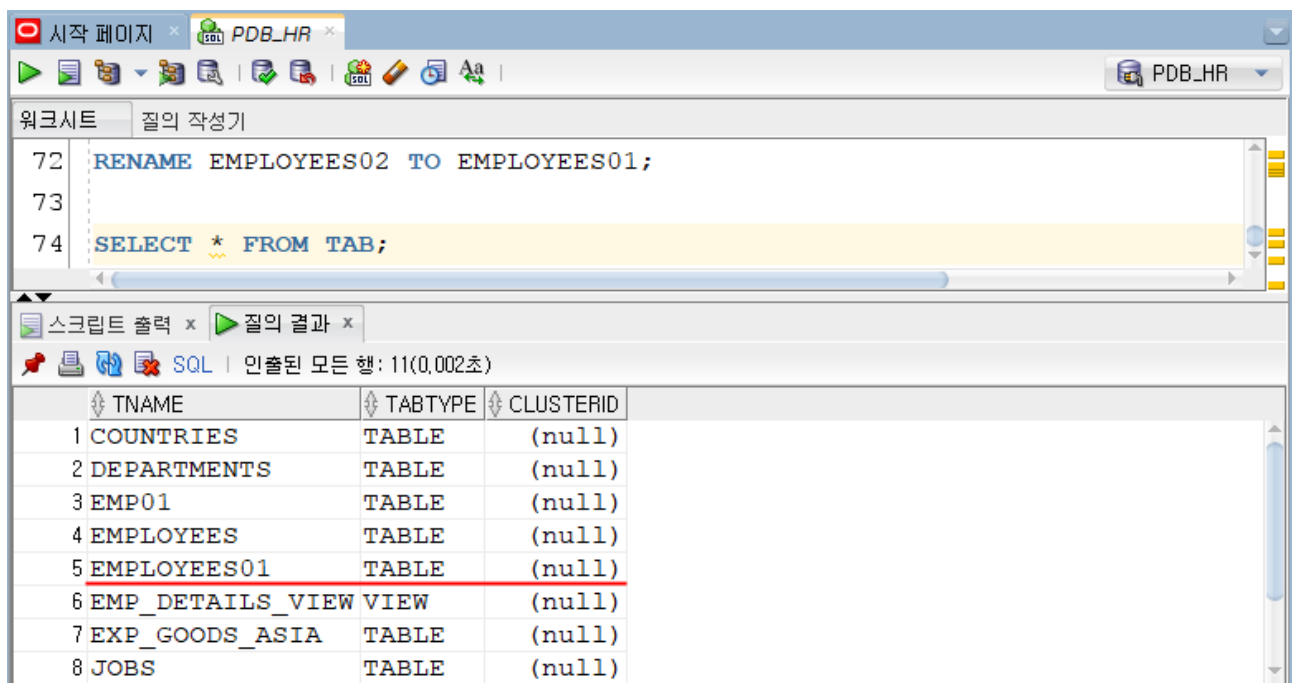
#### 5) 테이블 명을 변경하는 RENAME 문

**RENAME old\_name TO new\_name**

<예> EMPLOYEES02테이블의 이름을 EMPLOYEES01으로 변경

```
RENAME EMPLOYEES02 TO EMPLOYEES01;
```

```
SELECT * FROM TAB;
```



The screenshot shows a SQL IDE window titled 'PDB\_HR'. The SQL editor contains two statements: `RENAME EMPLOYEES02 TO EMPLOYEES01;` and `SELECT * FROM TAB;`. The results pane displays a table with the following data:

|   | TNAME            | TABTYPE | CLUSTERID |
|---|------------------|---------|-----------|
| 1 | COUNTRIES        | TABLE   | (null)    |
| 2 | DEPARTMENTS      | TABLE   | (null)    |
| 3 | EMP01            | TABLE   | (null)    |
| 4 | EMPLOYEES        | TABLE   | (null)    |
| 5 | EMPLOYEES01      | TABLE   | (null)    |
| 6 | EMP_DETAILS_VIEW | VIEW    | (null)    |
| 7 | EXP_GOODS_ASIA   | TABLE   | (null)    |
| 8 | JOBS             | TABLE   | (null)    |

#### 6) 테이블의 모든 로우를 제거해 TRUNCATE 문

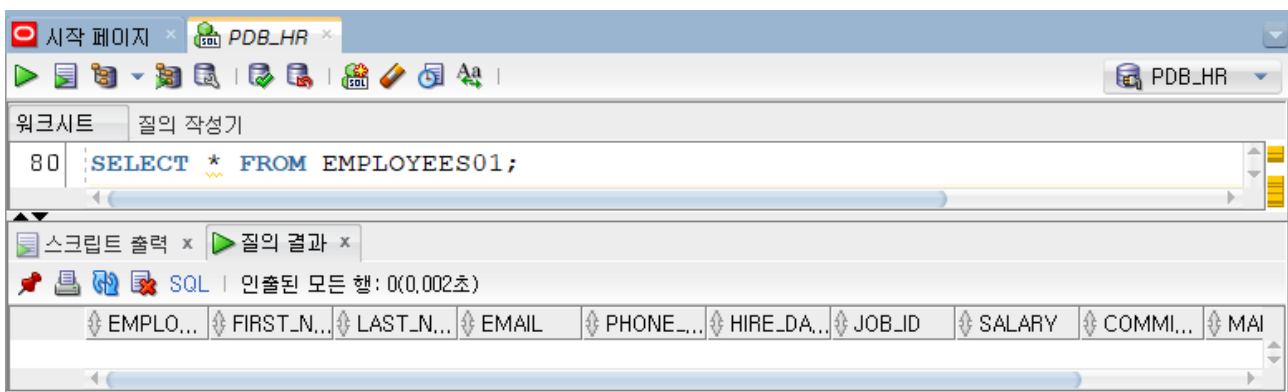
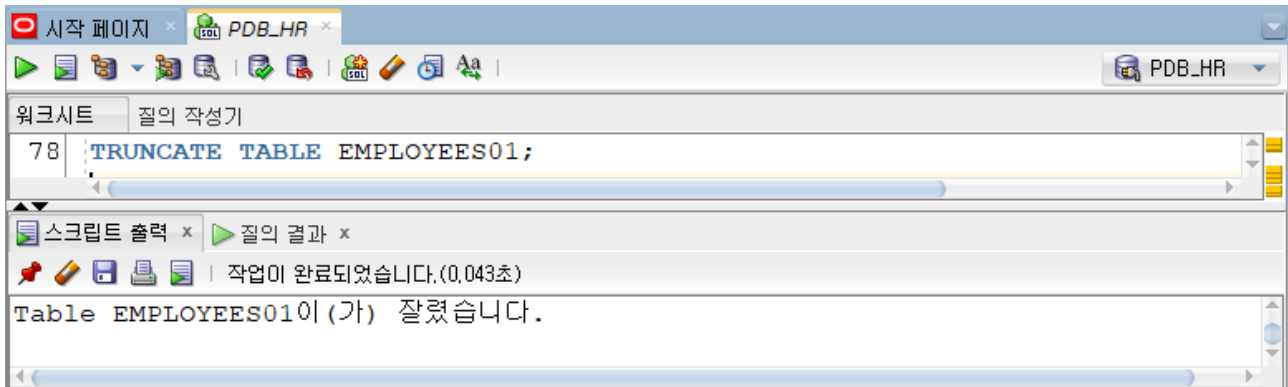
**TRUNCATE table table\_name**

<예> EMPLOYEES01테이블의 모든 로우를 제거

```
SELECT * FROM EMPLOYEES01;
```

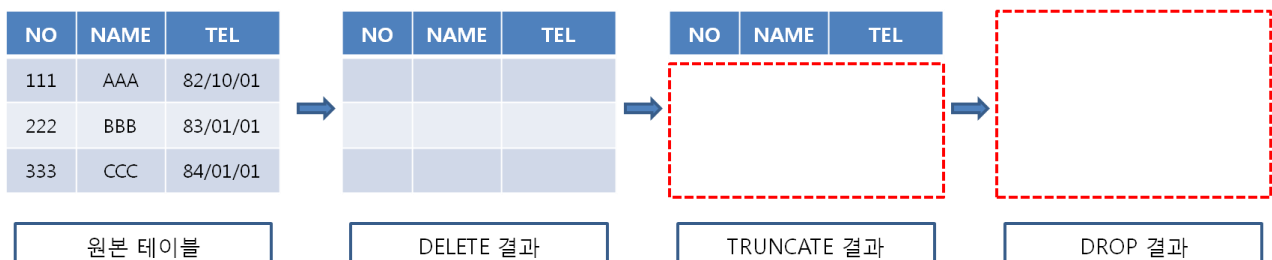
```
TRUNCATE TABLE EMPLOYEES01;
```

```
SELECT * FROM EMPLOYEES01;
```



- 테이블을 Truncate하면 테이블의 모든 행이 삭제되고 사용된 공간이 해제된다.
- TRUNCATE TABLE은 DDL명령이므로 롤백 데이터가 생성되지 않는다.  
DELETE명령으로 데이터를 지우면 롤백 명령어로 복구 할 수 있지만 TRUNCATE로 데이터를 삭제 하면 롤백을 할 수가 없다.
- 외래키가 참조중인 테이블은 TRUNCATE 할 수 없다.
- TRUNCATE명령을 사용하면 삭제 트리거가 실행되지 않는다.








※ DELETE, TRUNCATE, DROP 명령어의 차이점 비교



|          |                                      |
|----------|--------------------------------------|
| DELETE   | 데이터만 지워지고 쓰고 있던 디스크상의 공간을 그대로 가지고 있다 |
| TRUNCATE | 모든 데이터를 삭제하고 디스크상의 공간도 도 줄어들게 된다.    |
| DROP     | 데이터와 테이블 전체를 삭제하게 된다.                |

<실습>

- 고객 테이블 생성을 위한 필드(속성을 부여한 데이터) 설정

|  |       |                                      |
|--|-------|--------------------------------------|
|  고객코드<br>2017042        | 고객코드  | 고정 문자열 7자리, 공백 허용하지 않음, 기본키 적용       |
|  고객명<br>강원진             | 고객명   | 가변 문자열 10자리, 공백 허용하지 않음              |
|  성별<br>M                | 성별    | 고정 문자열 1자리, 공백 허용하지 않음, M(남성), W(여성) |
|  생일<br>19810603         | 생일    | 고정 문자열 1자리, 공백 허용하지 않음               |
|  전화번호<br>010-8202-7496  | 전화번호  | 가변 문자열 16자리, 공백 허용                   |
|  이메일<br>wjgang@navi.com | 이메일   | 가변 문자열 30자리, 공백 허용                   |
|  누적포인트<br>283500      | 누적포인트 | 숫자 10자리, 소수점 이하 없음                   |

- 고객 테이블 구조

테이블명: TB\_CUSTOMER. 기본키 설정방법: 컬럼명 자료형 PRIMARY KEY

| 필드명   | 영문 필드명       | 형식       | 크기 | NULL허용   | 기본키 | 비고             |
|-------|--------------|----------|----|----------|-----|----------------|
| 고객코드  | CUSTOMER_CD  | CHAR     | 7  | NOT NULL | PK  | PRIMARY KEY 설정 |
| 고객명   | CUSTOMER_NM  | VARCHAR2 | 20 | NOT NULL |     |                |
| 성별    | MW_FLG       | CHAR     | 1  | NOT NULL |     | M(남성) W(여성)    |
| 생일    | BIRTH_DAY    | CHAR     | 8  | NOT NULL |     |                |
| 전화번호  | PHONE_NUMBER | VARCHAR2 | 16 |          |     |                |
| 이메일   | EMAIL        | VARCHAR2 | 30 |          |     |                |
| 누적포인트 | TOTAL_POINT  | NUMBER   | 10 |          |     |                |
| 등록일시  | REG_DTTM     | CHAR     | 14 |          |     |                |

```

CREATE TABLE TB_CUSTOMER (
  CUSTOMER_CD      CHAR(7)          PRIMARY KEY,    -- 고객코드
  CUSTOMER_NM      VARCHAR2(10)     NOT NULL,         -- 고객명
  MW_FLG           CHAR(1)          NOT NULL,         -- 성별구분
  BIRTH_DAY        CHAR(8)          NOT NULL,         -- 생일
  PHONE_NUMBER     VARCHAR2(16),     -- 전화번호
  EMAIL            VARCHAR2(30),     -- EMAIL
  TOTAL_POINT      NUMBER(10),       -- 누적포인트
  REG_DTTM         CHAR(14 BYTE)     -- 등록일
);

```