00. SQL(Structured Query Language)문 살펴보기

- 관계형 데이터베이스에서 사용하는 표준 질의언어를 말한다.
- 사용방법이나 문법이 다른 언어(Java, C, C#)보다 단순하다.
- 모든 DBMS에서 사용 가능하다.
- 인터프리터 언어이다.
- 대소문자를 구별하지 않는다. (상수제외)

1. 데이터정의언어 - DDL(Data Definition Language)

데이터베이스의 테이블과 같은 데이터 구조를 정의하는데 사용되는 명령어들로
(생성, 변경, 삭제, 이름변경) 데이터 구조와 관련된 명령어들을 말함.

SQL문	내용
CREATE	데이터베이스 객체를 생성 한다.
DROP	데이터베이스 객체를 삭제 한다.
ALTER	기존에 존재하는 데이터베이스 객체를 다시 정의하는 역할을 한다.
RENAME	기존에 존재하는 테이블명을 변경한다.
TRUNCATE	테이블 또는 테이블의 지정된 파티션에서 모든 행을 제거한다.

2. 데이터조작언어 - DML(Data Manipulation Language)

스키마객체의 데이터를 입력, 수정, 조회, 삭제한다.

SQL문	내용
INSERT	데이터베이스 객체에 데이터를 입력한다.
DELETE	데이터베이스 객체의 데이터를 삭제한다.
UPDATE	기존에 존재하는 데이터베이스 객체의 데이터 수정한다.
SELECT	데이터베이스 객체로부터 데이터를 검색한다.

3. 데이터제어언어 - DCL(Data Contol Language)

권한의 설정과 회수

SQL문	내용
GRANT	데이터베이스 사용자 권한 설정
REVOKE	데이터베이스 사용자 권한 회수

4. 데이터제어언어 - TCL(Transaction Contol Language)

트랜잭션을 처리할수 있다.

SQL문	내용
COMMIT	데이터베이스 트랜잭션의 내용의 업데이트를 영구적으로 확정한다.
ROLLBACK	데이터베이스에서 업데이트가 오류가 발생할 때, 이전상태로 되돌리는 것을 말한다.
SAVEPOINT	특정부분에서 트랜잭션을 취소시킬수 있다.

4. 데이터 조작어(DML) (2)[SELECT문 및 연산자]

4.2 SELECT문 및 연산자

SELECT문은 데이터베이스로부터 저장되어 있는 데이터를 검색 하는데 사용합니다.

```
****** [Syntax] ***************************************************************************
*       SELECT [DISTINCT] {*, column [alias], ...}
*       FROM table_name
*       [WHERE conditon]
*       [ORDER BY {column, expression} [ASC | DESC]];
*******************************************************************************************
```

- DISTINCT : 중복되는 행을 제거 하는 옵션입니다.
- * : 테이블의 모든 column을 출력 합니다.
- alias : 해당 column에 대해서 다른 이름을 부여할 때 사용합니다.
- table_name : 질의 대상 테이블명
- WHERE : 조건을 만족하는 행들만 검색
- condition : column, 표현식, 상수 및 비교 연산자
- ORDER BY : 질의 결과 정렬을 위한 옵션(ASC:오름차순(Default), DESC:내림차순)

** SQL문의 작성 방법 **

- SQL 문장은 대소문자를 구별하지 않습니다.
- SQL 문장은 한 줄 또는 여러 줄에 입력될 수 있습니다.
- 일반적으로 키워드는 대문자로 입력합니다.
  다른 모든 단어, 즉 테이블 이름, 열 이름은 소문자로 입력합니다.(권장)
- 가장 최근의 명령어 1개가 SQL buffer에 저장 됩니다.
- SQL문 마지막 절의 끝에 " ; "를 기술하여 명령의 끝을 표시 합니다.

사용예)

```
SQL> SELECT emp_no as 사번, first_name 성명, gender FROM employees WHERE gender = 'M';
```

사번	성명	gender
10001	Georgi	M
10003	Parto	M
10004	Chirstian	M
10005	Kyoichi	M
10008	Saniya	M
10012	Patricio	M

emp_no와 first_name 은 각각 사번과 성명이라는 컬럼 별칭(alias)으로 만들어 출력했습니다.
alias를 사용할 때 as라는 키워드를 사용해도 되고, 생략할수도 있습니다.

** 오라클 연산자

1. 산술 연산자 : +, -, *, /
2. 비교연산자 : =, !=, <>, ^=, >, <, >=, <=
3. 논리 연산자 : AND 또는 && , OR 또는 || , NOT
4. WHERE절에 사용될 수 있는 SQL 연산자

연산자	설명
BETWEEN a AND b	a와 b 사이의 데이터를 출력한다. (a,b값 포함)
IN (list)	list의 값 중 어느 하나와 일치하는 데이터를 출력한다.
LIKE	문자 형태로 일치하는 데이터를 출력한다.(%, _사용)
IS NULL	NULL값을 가진 데이터를 출력한다.
NOT BETWEEN a AND b	a와 b사이에 있지 않은 데이터를 출력한다. (a, b값 포함하지 않음)
NOT IN (list)	list의 값과 일치하지 않는 데이터를 출력한다.
NOT LIKE	문자 형태와 일치하지 않는 데이터를 출력한다.
IS NOT NULL	NULL값을 갖지 않는 데이터를 출력한다.

** IN, NOT IN 연산자 **

IN 연산자

연산자 OR연산의 결과를 보여준다.

```
SQL> SELECT emp_no, first_name FROM employees
       WHERE emp_no IN(10005, 10009);--> 사번이 10005번, 10009번인 사원의 사번과 성명 출력
```

emp_no	first_name
10005	Kyoichi
10009	Sumant

NOT IN 연산자

```
SQL> SELECT emp_no, first_name FROM employees
       WHERE emp_no NOT IN (10003, 10005);--> 사번이 10005, 10009번이 아닌 사원의
                                              사번과 성명 출력
```

emp_no	first_name
10001	Georgi
10002	Bezalel
10004	Chirstian
10006	Anneke
10007	Tzvetan

** BETWEEN 연산자 **

연산자(AND를 이용해 두 조건을 결합한 검색과 같은 결과값을 보여줍니다.)

```
SQL> SELECT emp_no, first_name  FROM employees
     WHERE emp_no BETWEEN 30000 AND 30005 ; -> 급여가 3000에서 5000사이인 사원만 선택
```

emp_no	first_name
30000	Matt
30001	Izaskun
30002	Branimir
30003	Takahito
30004	Lucian
30005	Ramachenga

[문제] 사원테이블에서 사원명, 입사일, 성별을 선택한다.
단, 사원번호가 40001~40010사이인 사원의 레코드만 선택하라.

** LIKE 연산자 **

- 검색 STRING 값에 대한 와일드 카드 검색을 위해서 LIKE연산자를 사용 합니다.
- % : 여러개의 문자열을 나타내는 와일드 카드
- _ : 단 하나의 문자 를 나타내는 와일드 카드
- **ESCAPE** : 와일드 카드 문자를 일반문자 처럼 사용하고 싶은 경우 사용합니다.

```
WHERE name LIKE '%a₩_y%' ESCAPE '₩' ;
```

구분	설명
LIKE 'A%'	컬럼이 'A'로 시작하는 데이터들만 검색한다.
LIKE '%A'	컬럼이 'A'로 끝나는 데이터들만 검색한다.
LIKE '%KIM%'	컬럼에 'KIM' 문자가 있는 데이터들만 검색한다.
LIKE '%K%I%'	컬럼에 'K'문자와 'I'문자가 있는 데이터들만 검색한다.
LIKE '_A%'	컬럼에 'A'문자가 두 번째 위치한 데이터들만 검색한다.

※ '%'를 이용한 LIKE검색

```
SQL> SELECT emp_no, first_name  FROM employees  WHERE first_name LIKE '%K%' ;
```

emp_no	first_name
10005	Kyoichi
10006	Anneke
10010	Duangkaew
10016	Kazuhito
10018	Kazuhide
10020	Mayuko
10028	Domenick

※ '_'를 이용한 LIKE검색

```
SQL> SELECT emp_no, first_name
       FROM  employees
       WHERE  first_name like '_I%';
```

emp_no	first_name
10019	Lillian
10027	Divier
10043	Yishay
10044	Mingsen
10050	Yinghua
10051	Hidefumi

※ '_'는 한 문자를 나타냅니다.
'I' 문자가 두 번째 문자에 위치한 사원들의 정보를 보여줍니다.

[문제] 5월에 입사한 사원의 사원번호, 사원명, 입사일을 선택하라.

**** ORDER BY(ASC[오름차순], DESC[내림차순]) ****

ORDER BY 절은 데이터의 정렬을 위해 사용합니다.

```
SQL> SELECT emp_no, first_name
     FROM employees
     WHERE gender = 'F' && first_name like '_T%'
     ORDER BY emp_no ASC;
```

emp_no	first_name
10342	Stella
10403	Atreyi
10456	Stepehn
10661	Ottavia
11228	Utz

```
SQL> SELECT emp_no, first_name
     FROM employees
     WHERE gender = 'F' && first_name like '_T%'
     ORDER BY emp_no 1;
```

위 두 개의 쿼리는 동일한 결과를 가져 옵니다.

[연습문제]

1. EMPLOYEES테이블의 레코드 중 사원번호, 사원명, 입사일, 성별을 선택하라.

2. EMPLOYEES테이블의 레코드 중 사원번호, 사원명, 입사일을 선택하라.
   (단, 사원명을 오름차순으로 정렬하여 선택)

3. EMPLOYEES테이블의 레코드 중 6월, 12월에 입사한 사원의 사원번호, 사원명, 입사일을 선택하라.

4. EMPLOYEES테이블의 레코드 중 사원명에 'A'로 시작하고 생일이 1950년~1959년 사이인 사원을 나이가 많은 순으로 선택하라.

5. EMPLOYEES테이블의 레코드 중 1960년도에 태어난 남자사원을 선택하라.

6. EMPLOYEES테이블의 사원 중 1월에 태어난 여자 사원을 이름을 오름차순으로 선택하라.

7. EMP테이블의 사원 입사일이 1990-01-01이후인 사원과 이름에 B가 포함된 사원을 입사일 기준 내림차순으로 정렬하여 선택하라.

8. 현재 데이터베이스의 테이블 목록을 확인하는 쿼리문을 작성하라.

9. EMPLOYEES테이블의 사원 중 사원번호, 이름, 생년월일, 입사일을 이름은 오름차순, 사원번호는 내림차순으로 정렬하여 선택하라

10. EMPLOYEES테이블의 구조를 확인하는 쿼리문을 작성하라.

11. 현재 계정의 데이터베이스의 목록을 확인하는 쿼리문을 작성하라.

MySQL은 다양한 내장 함수를 포함하고 있습니다. 종류는 제어 흐름 함수, 문자열 함수, 수학 함수, 날짜/시간 함수, 전체 테스트 검색 함수, 형 변환 함수, XML 함수, 비트 함수, 보안/압축 함수 , 정보 함수, 공간 분석 함수, 기타 함수 등이 있습니다.

5. 내장 함수(1) [숫자함수]

5.1 숫자함수(Number Functions)

**** ABS(n) ****

ABS함수는 절대값을 계산하는 함수입니다.

```
SQL> SELECT ABS(-10) Absolute;

Absolute
--------
      10
```

**** CEIL(n), CEILING(n) - 올림 ****

CEIL함수는 주어진 값보다는 크지만 가장 근접하는 최소값을 구하는 함수 입니다.

```
SQL> SELECT CEIL10.1) TEST, CEILING(10.1) TEST2;

  TEST      TEST2
------- -----------
     11          11

SQL> SELECT CEIL(-10,1) TEST;

   TEST
-------
    -10
```

**** MOD(m,n), %, MOD - 나머지****

```
SQL> SELECT MOD(14, 3)  M1, 14 % 3 M2, 14 MOD 3 M3;

     M1         M2         M3
     2          2          2
```

**** FLOOR(n) - 내림 ****

FLOOR함수는 주어진 값보다 작거나 같은 최대 정수값을 구하는 함수입니다.

CEIL 함수와 비교해 보세요

```
SQL> SELECT FLOOR(10.1) TEST;

  TEST
-------
     10

SQL> SELECT FLOOR(-10.1) TEST;

   TEST
--------
      -11
```

**** RAND – 0~1사이의 난수 ****

LN함수는 주어진 값의 자연로그 값을 변환합니다.

```
SELECT RAND(), FLOOR(1 + (RAND() * 6));
```

**** POW(m, n), SQRT(n) - 거듭제곱, 제곱근 ****

POWER함수는 m의 n승 값을 계산합니다.

```
SQL> SELECT POW(4,2) TEST, SQRT(16) S;

      TEST          S
----------------------
        16          4
```

**** ROUND(n, [m]) ****

ROUND함수는 n값의 반올림을 하는 함수로 m(양수일 경우 소수이하, 음수일 경우 정수부분)은 소숫점 아래 자릿수를 나타낸다.

```
SQL> SELECT ROUND(191.123,1) TEST;    -- 소수이하 첫 번째 자리로 반올림

      TEST
---------
    191. 1

SQL> SELECT ROUND(192.123, -1) TEST; -- 정수부분 첫번째 자리를 두번째 자리로 반올림

      TEST
---------
      190
```

** SIGN(n) - SIGN은 숫자가 양수면 1 음수면 -1을 반환합니다. 0이면 0을 반환합니다.

```
SQL> SELECT SIGN(-10);
```

** TRUNC(n, m) **

TRUNC함수는 n값을 m 소숫점 자리로 반내림한 값을 반환합니다.(소수이하자리에 대한 버림)

ROUND함수와 비교해 보세요

```
SQL> SELECT TRUNC(7.5597, 2) TEST;

      TEST
---------
       7.55

SQL> SELECT TRUNC(5254.26, -2) TEST;

      TEST
---------
       5200
```

** FORMAT(숫자, 소숫점 자릿수) **

숫자를 소숫점 아래 자릿수까지만 표현합니다. 그리고 1000 단위마다 콤마를 표시합니다. 위에서는

```
SELECT FORMAT(123.1234, 2); -- FORMAT(숫자, 소숫점 자릿수) -->123.12
```

** CONV(숫자, 기존진수, 변환진수) **

CONV는 기존 진수의 숫자를 변환할 지수로 계산 후 반환합니다. 위에서는 10진수 100을 2진수로 변환한 144가 반환됩니다.

```
SQL> SELECT CONV(100, 10, 2) TEST;

      TEST
---------
       144
```

** BIN(), HEX(), OCT() : 2진수, 16진수, 8진수를 구한다. **

```
SELECT BIN(31), -- BIN(숫자)
       HEX(31), -- HEX(숫자)
       OCT(31); -- OCT(숫자)
```

5. 내장함수(2) [문자열 처리 함수]

5.2 문자열 처리 함수(Charaacter Functions)

** ASCII(s) **

ASCII는 문자의 아스키 코드값을 반환한다.

```
SQL> SELECT ASCII('A') NAME;
```

** CHAR(65) **

CHAR은 아스키 코드값에 해당하는 문자를 반환한다.

```
SQL> SELECT CHAR(65) NAME;
```

** BIT_LENGTH('TEST') - BIT크기 **

** CHAR_LENGTH('TEST') - 문자수 **

** LENGTH('TEST') - 할당된 BYTE 크기 **

```
SELECT BIT_LENGTH('abc'), CHAR_LENGTH('abc'), LENGTH('abc'); -- 24, 3, 3
SELECT BIT_LENGTH('가나다'), CHAR_LENGTH('가나다'), LENGTH('가나다'); -- 72, 3, 9
                                                    -->UTP-8 코드에서는 한글은 문자당 3바이트
```

** CONCAT(char1, char2,...) **

CONCAT은 문자열을 이을 때 사용합니다. 위에서는됩니다.

CONCAT_WS는 구분자와 함께 문자열을 이을 때 사용합니다. 위에서는 '2020/01/01'이 반환됩니다.

```
SQL>SELECT CONCAT('2020', '01', '01');   -- CONCAT(문자열1, 문자열2, ...)  -->'20200101'이 반환
SQL>SELECT CONCAT_WS('/','2020', '01', '01'); -- CONCAT_WS(구분자, 문자열1, 문자열2, ...)
                                                --> '2020/01/01'
```

** ELT(위치, 문자열, 문자열,...) : 위치 번째의 문자를 반환 **

** FIELD(찾을문자열, 문자열, 문자열,....) : 찾을 문자열의 위치를 찾아서 있으면 위치를, 없으면 0을 반환

** FIND_IN_SET(찾을문자열, '문자,문자,문자') : 찾을 문자열을 문자열 리스트에서 찾아서 위치를 반환합니다. 문자열 리스트는 콤마(,)로 구분되어 있어야 하며 공백이 없어야 한다.

** INSTR(기준문자열, 부분문자열) : 기준 문자열에서 부분 문자열을 찾아서 그 시작 위치를 반환.

** LOCATE(부분문자열, 기준문자열) : INSTR와 동일하지만 파라미터의 순서가 반대로 되어있습니다.

```
SQL> SELECT ELT(2, 'a', 'b', 'c'), -- ELT(위치, 문자열1, 문자열2, ...)              b
        FIELD('b', 'a', 'b', 'c'), -- FIELD(찾을 문자열, 문자열1, 문자열2, ...)          2
        FIND_IN_SET('b', 'a,b,c'), -- FIND_IN_SET(찾을 문자열, 문자열 리스트)      2
        INSTR('abcd', 'b'), -- INSTR(기준 문자열, 부분 문자열)          2
        LOCATE('b', 'abcd'); -- LOCATE(부분 문자열, 기준 문자열);        2
```

** INSERT(char, N, M, char) **

기준 문자열의 위치부터 길이만큼을 삽입할 문자열로 변경한다.

```
SQL> SELECT INSERT('가나다라마', 2, 3, '@@@'); -- '가@@@마'가 반환
```

** REVERSE(문자열) **

문자열을 거꾸로 만든다.

```
SQL> SELECT REVERSE('가나다라마'); -- '마라다나가'가 반환
```

** LEFT(문자열, n) : LEFT는 문자열의 왼쪽부터 길이만큼 반환**

** RIGHT(문자열, n) : RIGHT는 문자열의 오른쪽부터 길이만큼 반환**

```
SQL> SELECT LEFT('가나다라마바', 3), -- LEFT(문자열,길이) --> '가나다'가 반환
        RIGHT('가나다라마바', 3); -- RIGHT(문자열,길이) -->  '라마바'가 반환
```

** LCASE : 대문자를 소문자로 변경한 후 반환

** UCASE : 소문자를 대문자로 변경한 후 반환

```
SQL> SELECT LCASE(aBcDe), -- LCASE(문자열)    -->  'abcde'가 반환
            UCASE(aBcDe); -- UCASE(문자열)    -->  'ABCDE'가 반환
```

** LPAD(char1, n [,char2]) **

왼쪽에 문자열을 끼어 넣는 역할을 합니다. n은 반환되는 문자열의 전체 길이를 나타내며, char1의 문자열이 n보다 클 경우 char2을 n개 문자열 만큼 반환합니다.

```
SQL> SELECT LPAD('JUNG-SICK', 10, '*') NAME;

           NAME
--------------------
         *JUNG-SICK
```

** RPAD(char1, n [,char2]) **

LPAD와 반대로 오른쪽에 문자열을 끼어 넣는 역할을 합니다.

```
SQL> SELECT RPAD('JUNG-SICK', 10, '*') NAME;   --    JUNG-SHIK*
```

** SUBSTRING(char, m ,[n]) **

시작 위치부터 길이만큼 문자를 반환.

```
SQL> SELECT SUBSTRING('abcdef', 3, 2) NAME;    --    'cd'를 반환
```

** SUBSTRING_INDEX(문자열, 구분자 ,횟수) **

```
SQL> SELECT SUBSTRING_INDEX('jaehoney.tistory.com', '.', 2); --  'jaehoney.tistory'
SQL> SELECT SUBSTRING_INDEX('jaehoney.tistory.com', '.', -2); --    'tistory.com'
```

[문제] 이름을 글자길이의 50%만큼 출력하고 나머지 문자는 '*'으로 표시하여라.

**** REPEAT(문자열, 횟수) ****

문자열을 횟수만큼 반복

```
SQL> SELECT REPEAT('abc', 3) ; -- abcabcabc반환
```

**** REPLACE(char1, str1, str2) ****

REPLACE는 문자열의 특정 문자를 다른 문자로 변환 합니다.

```
SQL> SELECT REPLACE ('JACK and JUE', 'J', 'BL') "Changes";

Changes
--------------
BLACK and BLUE

SQL> SELECT REPLACE('JACK and JUE','JA','BL') "Changes";
Changes
------------
BLCK and JUE
```

-- 대소문자를 구분한다는 것을 알수 있습니다.

```
SQL>SELECT REPLACE('JACK and JUE','j','BL') "Changes";
Changes
------------
JACK and JUE
```

** LTRIM , RTRIM, TRIM**

특정한 문자를 제거 합니다.
LTRIM/RTRIM은 문자열의 왼쪽/오른쪽 공백을 제거합니다.

TRIM은 양쪽의 공백을 모두 제거 합니다.

```
SQL> SELECT LTRIM('    abc'), -- LTRIM(문자열)
        RTRIM('abc    '),   -- RTRIM(문자열)
        TRIM('   abc   ');   -- TRIM(문자열)

SQL> SELECT TRIM(BOTH 'a' FROM 'aababaa') "TRIM Example";   --   'bab'를 반환
```

[문제]이메일을 이용하여 아이디와 도메인을 분리하여라.

5. 내장함수(3) [날짜 처리 함수]

5.3 날짜 처리 함수(Date Functions)

날짜 -> YYYY, YY - 년도, MM -월, DD-일, DY, DAY -요일(월, 월요일)

시간 -> HH(12시기준) HH24(24시기준) - MI(분) - SS(초) PM(AM,PM)

** 현재 시스템의 날짜와 시간을 구하는 함수이다.

CURDATE() : 현재 년-월-일을 반환

CURTIME() : 현재 시:분:초을 반환

NOW() : 년-월-일 시:분:초을 반환

SYSDATE() : 년-월-일 시:분:초을 반환

DATE() : 날짜와 시간에서 년-월-일을 반환

TIME() : 날짜와 시간에서 시:분:초을 반환

```
SQL> SELECT CURDATE(), CURTIME(), NOW(), SYSDATE();
SQL> SELECT  DATE(NOW()), TIME(NOW());
```

** 날짜데이터에서 특정 년, 월, 일, 시, 분, 초, 밀리초를 구한다.

```
SQL>  SELECT  YEAR(NOW()),  MONTH(NOW()),  DAYOFMONTH(NOW()),  HOUR(NOW()),
              MINUTE(NOW()), SECOND(NOW()), MICROSECOND(NOW());
```

** ADDDATE : 날짜를 기준으로 차이를 더한 날짜를 반환한다.

** SUBDATE : 날짜를 기준으로 차이를 뺀 날짜를 반환한다.

```
SQL> SELECT ADDDATE('2020-01-01', INTERVAL 31 DAY),
                                          -- ADDDATE(날짜, 차이)  '2020-02-01'
            SUBDATE('2020-01-01', INTERVAL 31 DAY);
                                          -- SUBDATE(날짜, 차이) '2019-12-01'
```

** DATEDIFF(날짜1, 날짜2) : 날짜2에서 날짜1까지 몇 일 남았는지를 반환한다.

** TIMEDIFF(날짜1 OR 시간1, 날짜2 OR 시간2) : 시간이 얼마나 남았는지를 반환

```
SQL> SELECT DATEDIFF('2020-1-5', '2020-1-15'),      --  4
            TIMEDIFF('14:30:00', '06:30:00');       --  08:00:00
```

** DAYOFWEEK(날짜) : 요일을(월:2 화:3) 반환한다.

** MONTHNAME(날짜) : 해당 월의 영어이름 반환

** DAYOFYEAR(날짜) : 1년 중 몇일이 지났는지를 반환

```
SQL> SELECT DAYOFWEEK(NOW()),     -- 2
            MONTHNAME(NOW()),  -- JANUARY
            DAYOFYEAR(NOW());  -- 3
```

** LAST_DAY(날짜) : 주어진 월의 마지막날을 반환

```
SQL> SELECT LAST_DAY('2022-02-04');  -- 2022-02-28
```

** TIME_TO_SEC(시간) : 시간을 초 단위 반환

```
SQL> SELECT TIME_TO_SEC('5:53:10');  -- 21190
```

5. 내장함수(4) [변환 함수]

5.4 변환 함수 (Conversion Functions)

데이터형식 : BINARY, CHAR, DATE, DATETIME, SIGNED, TIME, UNSIGNED

**** CONVERT (표현할값, 데이터형식[(길)]) * ***

형 변환 함수는 위 두 가지이다. CAST와 CONVERT는 같은의미로, 표현할 값을 해당 데이터 형식으로 바꾸는 것이다。

데이터형식[(길이)]에서 [(길이)]는 생략가능함.

SIGNED INTEGER는 부호가 있는 정수(-21억~21억)

UNSIGNED INTEGER는 부호가 없는 정수(0~42억)

```
SQL> SELECT AVG(SALARY), CONVERT(AVG(SALARY), signed INTEGER) FROM SALARIES;
```

**** CAST (표현할값 AS 데이터형식[(길이)]) ****

```
SQL> SELECT CAST('2022~01~01' AS DATE),
            CAST('2022/01/01' AS DATE),
            CAST('2022.01.01' AS DATE),
            CAST('2022,01,01' AS DATE);

SQL> SELECT  CAST(NOW() AS DATE);
```

** DATE_FORMAT(날짜시간, 포맷형식) **

종류		역할
%Y	년	4자리 연도
%y	년	2자리 연도
%m	월	2자리 (00-12)
%c	월	1자리, 10보다 작을경우 (1-12)
%M	월	January, February, March, April, May, June, July, August, September, October, November, December
%b	월	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
%d	일	2자리 (00-31)
%e	일	1자리, 10보다 작을경우 (0-31)
%D	일	1st, 2nd...
%H	시	24시간 형식 (00-23)
%h	시	12시간 형식 (01-12)
%I	시	12시간 형식 (01-13)
%k	시	24시간 형식, 10보다 작을경우 한자리 (0-23)
%l	시	12시간 형식, 10보다 작을경우 한자리 (1-12)
%i	분	2자리 (00-59)
%S	초	2자리 (00-59)
%s	초	2자리 (00-59)
%f	마이크로초	100만분의 1초
%p	오전/오후	AM/PM
%T	시분초	24시간 형식 (hh:mm:ss)
%r	시분초 오전/오후	12시간 형식 (hh:mm:ss AM/PM)
%j	일	그해의 몇번째 일인지 표시 (001-366)
%w	일	그주의 몇번째 일인지 표시 (0=일요일, 6=토요일)
%W	주	Monday,Tuesday,Wednesday, Thursday, Friday, Saturday, Sunday
%a	주	줄인 이름(Mon,Tue, Wed, Thu, Fri, Sat, Sun)
%U	주	그해의 몇번째 주인지 표시 (00-53) 일요일이 주의 첫번째일
%u	주	그해의 몇번째 주인지 표시 (00-54) 월요일이 주의 첫번째일
%X	년	그주가 시작된 해을 표시, %V와 같이 사용
%x	년	그주가 시작된 해을 표시, %v와 같이 사용
%V	주	그주가 시작된 해의 몇번째 주인지 표시 (01-53) 일요일이 주의 첫번째일 %X 와 함께사용
%v	주	그주가 시작된 해의 몇번째 주인지 표시 (01-53) 월요일이 주의 첫번째일 %x 와 함께사용

```
SQL> SELECT DATE_FORMAT(NOW(), '%Y-%m-%d') ;     -- 2022-01-05
SQL> SELECT DATE_FORMAT('20220405','%Y/%m/%d');   -- 2022/04/05
SQL> SELECT DATE_FORMAT('2020-04-05', '%W %M %Y') ; -- Sunday April 2020
SQL> SELECTDATE_FORMAT(NOW(), '%Y-%m-%d %H:%i:%S %p %W %a');
```

	DATE_FORMAT(NOW(), '%Y-%m-%d %H:%i:%S %p %W %a')
▸	2022-01-05 14:21:05 PM Wednesday Wed

6. 그룹 함수(1) [Group Function의 종류]

6.1 Group Function의 종류

**** 그룹함수란? ****

- 그룹함수란 여러 행 또는 테이블 전체의 행에 대해 함수가 적용되어 하나의 결과값을 가져오는 함수를 말합니다.
- GROUP BY절을 이용하여 그룹 당 하나의 결과가 주어지도록 그룹화 할 수 있습니다.
- HAVING절을 사용하여 그룹 함수를 가지고 조건비교를 할 수 있습니다.
- COUNT(*)를 제외한 모든 그룹함수는 NULL값을 고려하지 않습니다.
- MIN, MAX 그룹함수는 모든 자료형에 대해서 사용 할 수 있습니다.

**** COUNT ****

COUNT 함수는 검색된 행의 수를 반환 합니다.

```
SQL> SELECT COUNT(deptno) FROM DEPT;

  COUNT(DEPTNO)
----------------
             4
```

검색된 행의 총 수 4개를 반환합니다. 즉 4개의 부서가 존재합니다.

**** MAX ****

MAX함수는 컬럼중의 최대값을 반환 합니다.

```
SQL> SELECT MAX(sal) salary FROM emp;

    SALARY
----------
      5000
```

sal컬럼중에서 제일 큰값을 반환합니다. 즉 가장 큰 급여를 반환합니다.

**** MIN ****

MIN 함수는 컬럼중의 최소값을 반환 합니다.

```
SQL> SELECT MIN(sal) salary FROM emp;

    SALARY
----------
       800
```

sal컬럼중에서 가장 작은 값 반환합니다. 즉 가장 적은 급여를 반환합니다

**** AVG ****

AVG함수는 평균값을 반환 합니다.

```
SQL> SELECT ROUND(AVG(sal), 1) salary FROM emp WHERE deptno = 30;

    SALARY
----------
    1566.7
```

30부서 사원의 평균 급여를 소수점 1자리 이하에서 반올림해서 보여줍니다.

**** SUM ****

SUM함수는 검색된 컬럼의 합을 반환 합니다.

```
SQL> SELECT SUM(sal) salary FROM emp WHERE deptno = 30;

    SALARY
----------
      9400
```

30부서 사원의 급여 합계를 보여줍니다.

6. 그룹함수(2) [Group By절과 Having절]

6.1 GROUP BY절과 HAVING절

**** GROUP BY절 ****

- 특정한 컬럼의 데이터들을 다른 데이터들과 비교해 유일한 값에 따라 무리를 짓습니다.
- GROUP BY절을 사용하여 한 테이블의 행들을 원하는 그룹으로 나눕니다.
- Column명을 GROUP함수와 SELECT절에 사용하고자 하는 경우 GROUP BY뒤에 Column명을 추가 합니다.

```
SQL> SELECT b.deptno, COUNT(a.empno)
       FROM emp a, dept b
       WHERE a.deptno = b.deptno
       GROUP BY b.deptno;
```

```
DEPTNO   COUNT(*)
-------  ----------
    10          3
    20          5
    30          6
```

부서별로 그룹을 지은 검색 결과 값이며

부서별로 사원수를 보여줍니다.

**** Group By 예제 ****
scott/tiger유저로 접속해서 실행하세요..

예제1) 부서별로 그룹하여 부서번호, 인원수, 급여의 평균, 급여의 합을 구하여 출력 하여라.

```
SQL> SELECT DEPTNO, COUNT(ENAME), AVG(SAL), SUM(SAL)
     FROM EMP
     GROUP BY DEPTNO
```

```
  DEPTNO   COUNT(*)   급여평균   급여합계
---------- ---------- ---------- ----------
    10          3        2998       8995
    20          5        2175      10875
    30          6        1567       9400
```

예제2)업무별로 그룹하여 업무, 인원수, 평균 급여액, 최고 급여액, 최저 급여액 및 합계를 출력하라.

```
SQL> SELECT job, count(empno) 사원수, avg(sal) 평균급여, min(sal) 최저급여,
            sum(sal) 급여합계, max(sal) 최고급여
     FROM emp
     GROUP BY job;
```

job	사원수	평균급여	최저급여	급여합계	최고급여
CLERK	4	1037.500000	800.00	4150.00	1300.00
SALESMAN	4	1400.000000	1250.00	5600.00	1600.00
MANAGER	3	2758.333333	2450.00	8275.00	2975.00
ANALYST	2	3000.000000	3000.00	6000.00	3000.00
PRESIDENT	1	5000.000000	5000.00	5000.00	5000.00

**** GROUP BY HAVING 절 ****

- WHERE절에 GROUP Function을 사용할 수 없습니다.
- HAVING절은 GROUP 함수를 가지고 조건비교를 할 때 사용 합니다
- **WHERE** -> **GROUP BY** -> **HAVING** -> **ORDER BY**순으로 쿼리문이 와야 됩니다.

**** HAVING절 예제 ****

예제1) 사원수가 5명이 넘는 부서의 부서명과 사원수를 출력해라

```
SQL>SELECT b.dname, COUNT(a.empno)
    FROM emp a, dept b
    WHERE a.deptno = b.deptno
    GROUP BY dname
    HAVING COUNT(a.empno) >5 ;
```

dname	COUNT(a.empno)
SALES	6

예제2) 전체 월급이 5000을 초과하는 각 업무에 대해서 업무와 월급여 합계를 출력하여라.
단 판매원은 제외하고 월 급여 합계로 내림차순 정렬 하여라.

```
SQL>SELECT job, SUM(sal) "급여합계"        --> 업무와 급여 합계를 출력
    FROM emp
    WHERE job NOT IN ('SALES')          --> 판매원은 제외
    GROUP BY job                        --> 업무별로 Group By
    HAVING SUM(sal) >5000               --> 전체 월급이 5000을 초과하는
    ORDER BY SUM(sal) DESC;             --> 월급여 합계로 내림차순 정렬
```

	job	SUM(sal)
▸	MANAGER	8275.00
	ANALYST	6000.00

4. 데이터 조작어 (DML)(1)[데이터의 삽입, 수정, 삭제]

4.1 데이터의 삽입, 수정, 삭제

** INSERT **

INSERT명령어는 테이블 안에 데이터를 삽입하는 역할을 합니다.

```
** [Syntax] ***********************************
*    INSERT INTO table_name(column1, column2,...)
*    VALUES (데이터, '데이터',...);
************************************************
```

- 실제 데이터는 VALUES 괄호()안에 입력하고 문자열은 단일 따옴표(' ')로 둘러 쌉니다.
- 각각의 데이터 구분은 " , "로 합니다.
- 테이블 이름 옆에 ()생략시에는 모든 컬럼을 VALUES()안에 입력 시킵니다.

** 모든 데이터를 입력할 경우 **

```
SQL> INSERT INTO emp
        VALUES(7302, 'SMITH', 'CHERK', 7939, '80/12/17', 80, NULL, 20);
```

** 원하는 데이터만 입력할 경우

```
SQL> INSERT INTO dept (deptno, dname)
          VALUES(10, 'ACCOUNTING' );
```

```
** SELECT 문장을 이용한 INSERT **********************
*    INSERT INTO table_name(column1, column2,...)
*    SELECT column1, column2,.....
*    FROM table_name
*    WHERE 조건 ;
*******************************************************
```

```
SQL> INSERT INTO dept
          SELECT * FROM dept;
```

** **UPDATE** **

테이블 안의 데이터를 수정 합니다.

** [Syntax] ***************************************

* **UPDATE** table_name

* **SET** column1 = 값(고칠내용), column2 = 값, ...

* **WHERE** 조건

****

```
SQL> UPDATE emp
        SET deptno = 30
        WHERE empno = 7902;
```

사원번호가 7902번인 사람의 부서 번호가 30번으로 수정됨

```
SQL> UPDATE emp
        SET sal = sal * 1.1
        WHERE deptno = 20;
```

20부서의 사원들의 급여가 10% 인상됨

```
SQL> UPDATE emp
        SET HIREDATE = now();
```

모든 사원의 입사일이 오늘로 수정됨

** **DELETE** **

사용하지 않는 데이터를 삭제 합니다.

** [Syntax] *******************************

* **DELETE FROM** table_name **WHERE** 조건 ;

****

```
SQL> DELETE FROM emp
        WHERE empno = 7902;
```

사원번호가 7902번인 사람의 데이터가 삭제 되었습니다.

```
SQL> DELETE FROM emp
        WHERE sal <(SELECT avg(sal) FROM emp);
```

평균급여보다 적게 받는 사원 삭제

```
SQL> DELETE FROM emp;
```

테이블의 모든행이 삭제 됩니다.

3. 테이블의 생성과 수정 그리고 삭제(4)[테이블의 관리]

3.5 테이블의 관리

테이블의 관리는 테이블의 컬럼 관리와 테이블 정보 관리로 나누어서 설명 합니다.

**** 테이블의 컬럼의 관리 ****

테이블의 컬럼은 ADD, MODIFY, DROP 연산자를 통해서 관리 할 수 있습니다.

1. ADD 연산자 : 테이블에 새로운 **컬럼을 추가** 할 때 사용 합니다.

** ADD 연산자 예제 **

SQL> ALTER TABLE emp **ADD** addr VARCHAR2(50);

VARCHAR2의 데이터 형을 가지는 addr 칼럼이 emp 테이블에 추가 됩니다.

2. MODIFY 연산자 : 테이블의 컬럼의 크기를 수정 하거나 NOT NULL컬럼으로 변경 할 수 있습니다.

** MODIFY 연산자 예제 **

SQL> ALTER TABLE emp **MODIFY** ename VARCHAR(50);

SQL> ALTER TABLE emp **MODIFY** ename VARCHAR(50) NOT NULL;

3. CHANGE 연산자 : 테이블의 컬럼명을 변경 할 수 있습니다.

** CHANGE 연산자 예제 **

ALTER TABLE emp2 CHANGE TEL TEL3 VARCHAR(20);

4. DROP 연산자 : 테이블 컬럼을 삭제 하거나, 테이블의 제약 조건을 삭제 할 때 사용 합니다.

** 컬럼의 삭제 예제 **

-- 컬럼의 삭제는 오라클 8i버전부터 지원을 합니다.

SQL> ALTER TABLE table_name **DROP COLUMN** column_name

** 테이블 정보의 관리 **

1. 기존 테이블의 복사

- 기존 테이블을 부분, 또는 완전히 복사할 때에 **서브쿼리를 가진 CREATE TABLE 명령어를** 사용해서 쉽게 테이블을 복사 할 수 있습니다.
- 하지만 제약 조건, 트리거, 그리고 테이블 권한은 새로운 테이블로 복사되지 않습니다.
- 제약조건은 NOT NULL 제약조건만 복사 됩니다.

```
** [Syntax] ******************************
* CREATE TABLE [ schema. ] table_name
*                  [ LOGGING | NOLOGGING ]
*                 [ ... ]
*                 AS
*                 subquery
*****************************************
```

** 예제 **

```
SQL> CREATE TABLE emp2
       AS
       SELETE * FROM emp;
```

테이블이 생성 되었습니다.

2. DROP TABLE(테이블을 삭제할 때 사용)

```
** [Syntax] ***************************************************
* DROP TABLE [ shema. ] table_name ;
*************************************************************
```

```
SQL> DROP TABLE emp;
```