8253jang@daum.net
장희정

문자열 함수

- upper(문자열) : 모두 대문자
- lower(문자열) : 모두 소문자

- length(문자열) : 문자열의 byte 길이
- char_length(문자열) : 문자열의 길이

- concat(문자열, 문자열,...) : 문자열과 문자열을 연결
  ex) SELECT CONCAT('I ', 'Love ', 'MySQL') ;

- substring(문자열, 시작, 개수) : 문자의 일부분 추출
  ex) substring(문자열, INDEX) : 문자열에서 INDEX 부터 끝까지 추출(index는 1부터 시작)
  substring(문자열, INDEX, 개수) : 문자열에서 INDEX 부터 개수 까지 추출

  ☞ substring()함수를 substr()로 해도 가능하다.
  substr함수는 substring 함수의 Alias로 완전히 동일한 기능.
  sql 표쥰은 substring 사용

- instr(문자열, 찾을문자열) : 문자열 내에서 특정 문자열이 처음 등장하는 위치(인덱스)를 반환
  만약, 찾는 문자열이 없으면 0 이다.

  ex) SELECT INSTR('Hello World', 'World'); -- 7

- locate(찾을문자열, 문자열) : 문자열 내에서 특정 문자열이 처음 등장하는 위치(인덱스)를 반환

  ☞ instr 함수와 locate 함수는 거의 동일하지만 매개변수 순서가 다르다.

-position('특정문자열' in '문자열') : 특정문자까지의 문자열 길이을 반환
  특정문자가 존재하지 않으면 0

  ☞ instr, locate, position은 모두 특정 문자열이 처음 등장하는 우치를 반환 한다.
  instr, locate 는 mysql의 고유함수, **postion 은 sql 표준**
  : locate는 MySQL, MariaDB
  : instr는 MySQL, Oracle

  ex) SELECT POSITION('d' IN 'abcd abcd abcd'); -- 결과 4

8253jang@daum.net
장희정

SELECT POSITION('f' IN 'abcd abcd abcd'); -- 결과 0

- left(문자열 , 길이) : 지정한 길이만큼 왼쪽부터 문자열 반환
- right(문자열 , 길이) : 지정한 길이만큼 오른쪽부터 문자열 반환

ex)

SELECT left('jang hee jung', 4); -- 결과 jang

SELECT right('jang hee jung', 4); -- 결과 jung

- ltrim() : 왼쪽 공백제거
- rtrim() : 오른쪽 공백제거
- trim() : 양쪽공백제거

- replace(문자열, 지정한문자 , 다른문자) : 지정한 문자를 다른 문자로 대체함.

ex) SELECT REPLACE('jang hee jung', 'j', 'k');

☞ 대소문자 가림.

- repeat('문자', 반복횟수) : 지정한 문자를 반복

- reverse() : 문자열을 거꾸로 만듦

- strcmp(문자열, 문자열) : 문자열 비교

STRCMP 함수는 두 문자열을 비교하여 동일한지를 알려 줌

비교하는 두 문자열이 동일할 경우 0을, 다를 경우 -1을 반환함

날짜 함수

CURRENT_date() : 서버의 현재 날짜

CURRENT_TIME () : 서버의 현재 현재시간

CURRENT_TIMESTAMP() : 서버의 현재 날짜와 시간

NOW() : 서버의 현재 날짜와 시간

date_add(날짜, INTERVAL 숫자 단위) : 날짜 더하기

date_sub(날짜, INTERVAL 숫자 단위) : 날짜 빼기

8253jang@daum.net
장희정

☞ DATE_ADD , DATE_SUB 함수는 첫 번째 인수로 날짜 데이터를 입력하고,
두 번째 인자로 INTERVAL과 함께 더하거나 빼고자 하는 숫자 그리고 연, 월, 일 등의 단위를 넣음. 더하거나 빼는 숫자와 함께 사용하는 단위 - YEAR ,QUARTER , MONTH, DAY, WEEK, HOUR , MINUTE,SECOND , MICROSECOND

ex)

```
SELECT NOW(), DATE_ADD(NOW(), INTERVAL 1 YEAR);
SELECT NOW(), DATE_ADD(NOW(), INTERVAL -1 YEAR);
SELECT NOW(), DATE_SUB(NOW(), INTERVAL 1 YEAR), DATE_SUB(NOW(), INTERVAL -1 YEAR);
```

datediff(시작날짜, 종료날짜) : 날짜 간의 일 수 차 를 구함 , 실행결과는 일수를 반환

timestampdiff() : 일수가 아닌 연 또는 시간 등 다양한 단위로 확인하고 싶다면 TIMESTAMPDIFF 함수

```
ex) SELECT DATEDIFF('2025-2-10', now()) ;
    SELECT TIMESTAMPDIFF(year, '2025-1-10', now() );
    SELECT TIMESTAMPDIFF(MONTH, '2025-1-10', now() );
    SELECT TIMESTAMPDIFF(day, '2025-1-10', now() );
    SELECT TIMESTAMPDIFF(hour, '2025-1-10', now() );
    SELECT TIMESTAMPDIFF(minute, '2025-1-10', now() );
    SELECT TIMESTAMPDIFF(second, '2025-1-10', now() );
```

dayname(날짜) : 지정한 날짜의 요일을 반환

```
SELECT DAYNAME(now());
```

year(날짜) : 날짜에서 년도 추출
month(날짜) : 날짜에서 월 추출
week(날짜) : 날짜에서 주 추출
day(날짜) : 날짜에서 일 추출

```
ex) SELECT YEAR(now()), MONTH(now()), WEEK(now()), DAY(now());
```

date_format() : DATE_FORMAT 함수는 날짜를 다양한 형식으로 표현해야 할 때 사용함.
나라마다 날짜를 표현하는 방식이 다르므로 날짜 형식으로 변환해야
할 때 DATE_FORMAT 함수가 필요함

8253jang@daum.net
장희정

ex) SELECT DATE_FORMAT(now() , '%m/%d/%Y');
SELECT DATE_FORMAT(now() , '%m/%d/%Y');
SELECT DATE_FORMAT(now(), '%Y%m%d');
SELECT DATE_FORMAT(now(), '%Y.%m.%d');
SELECT DATE_FORMAT(now(), '%H:%i:%s');

데이터타입 변환

숫자를 날짜로 또는 날짜를 숫자로 변환하는 등 데이터를 다양한 형태로 변환해야 하는 경우가 있다.

- cast()
  : SELECT CAST(열 AS 데이터 유형) FROM 테이블;

- convert()
  : SELECT CONVERT(열, 데이터 유형) FROM 테이블;

```
-- datetime을 정수형으로 변환
SELECT NOW(), CAST(NOW() AS SIGNED) , CAST(NOW() AS unSIGNED) ;

select cast(20250215231531 as date);
select cast(20250215231531 as datetime);

select cast('20250215231531' as date);
select cast('20250215231531' as datetime);

select cast('20250215231531' as char);
select cast('20250215231531' as char(4));
select cast('20250215231531' as char(8));

select convert(20250215231531 , date);
select convert(20250215231531 , datetime);

select convert('20250215231531' , date);
select convert('20250215231531' , datetime);

select convert('20250215231531' , char);
```

8253jang@daum.net
장희정

```
select convert('20250215231531' ,char(4));
select convert('20250215231531' , char(8));
```

NULL 관련 함수

- IFNULL(열, 대체할 값)
- COALESCE(열 1, 열 2, ...) : COALESCE는 NULL이 아닌 값이 나올 때까지 후보군의 여러 열을 입력할 수 있다.

```
-- NULLIF는 첫번째 인수와 두번째 인수가 같으면 NULL, 아니면 첫 번째 인수의 값
SELECT EMPNO, JOB, NULLIF(JOB, 'MANAGER')  FROM EMP;


-- COALESCE함수 ; 가장 먼저 NULL이 아닌 것을 반환
SELECT ENAME, COMM, SAL, COALESCE(COMM, SAL, 50) RESULT
FROM EMP;

  SELECT COALESCE(100, NULL,200, 300) FROM DUAL;
  SELECT COALESCE( NULL,100,200, 300) FROM DUAL;
  SELECT COALESCE(NULL, NULL,200, 300) FROM DUAL;
  SELECT COALESCE(NULL, NULL,NULL, 300) FROM DUAL;
```

조건함수

조건에 따라 값을 다르게 반환

```
1)  IF(condition,  true_value,  false_value)

2)  case [대상]
        when 조건1 then 문장
        when 조건2 then 문장
        when 조건3 then 문장
        ...
        else 문장
```

8253jang@daum.net
장희정

```
        end;
```

숫자함수

- round(숫자, 자리수) : 반올림

- ceiling(숫자) : 올림 한 후 정수반환

- floor(숫자) : 내림 한 후 정수 반환

- mod(나누어지는수, 나누는 수) : 나먼지 연산자 , %와동일

집계함수

집계함수는 데이터를 그룹화해 계산 할 때 자주 사용된다.

- sum(컬럼명) : 합계

- avg(컬럼명) : 평균(null값은 제외하고 나눔)

- max(컬럼명) : 최대값

- min(컬럼명) : 최소값

- count(컬럼명) : 총 레코드수(null값은 제외함)

- count(*) : null을 포함한 총 레코드수

집계함수 TEST를 위한 테이블

```
CREATE TABLE REPORT(
  NAME VARCHAR(20)   PRIMARY KEY,
  BAN CHAR(1),
  KOR INT CHECK(KOR BETWEEN 0 AND 100),
  ENG INT CHECK(ENG BETWEEN 0 AND 100),
  MATH INT CHECK(MATH BETWEEN 0 AND 100)
);

SELECT * FROM REPORT;

-- 샘플레코드
INSERT INTO REPORT VALUES('희정', 1 , 80, 70,90);
INSERT INTO REPORT VALUES('효리', 1 , 90, 50,90);
```

8253jang@daum.net
장희정

```
INSERT INTO REPORT VALUES('나영', 2 , 100, 65,85);
INSERT INTO REPORT VALUES('재석', 2 , 80, 70, 95);
INSERT INTO REPORT VALUES('희선', 2 , 85, 45,80);

INSERT INTO REPORT VALUES('승기', 3 , 50, 70,70);
INSERT INTO REPORT VALUES('중기', 3 , 90, 75,80);
INSERT INTO REPORT VALUES('혜교', 3 , 70, 90,95);
INSERT INTO REPORT VALUES('미나', 3 , NULL, 80,80);
```

집계함수를 이용할 때 group by에 없는 일반 컬럼명은 함께 검색할 수 없다.

일반컬럼과 집계함수를 함께 검색하려면 GROUP BY 절을 이용한다.

특정컬럼 기준으로 group by 를 하여 각 그룹별 집계함수를 사용 수 있다

집계함수를 조건으로 사용 할때는 HAVING 절을 사용해야 한다.

```
-- SELECT 실행 순서
SELECT    5)
FROM      1)
WHERE     2)
GROUP BY 3)
HAVING    4)
ORDER BY 6)
```

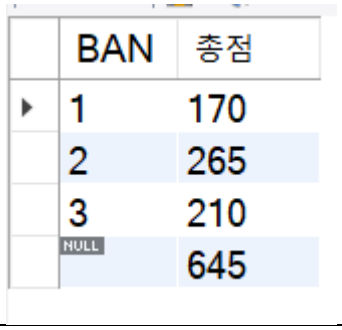
ROLLUP - 부분합과 총합을 구하는 함수
GROUP BY 문과 ROLLUP함수 조합
ex) GROUP BY 열이름 with ROLLUP

```
SELECT BAN , SUM(KOR) 총점 FROM REPORT GROUP BY BAN;
```

8253jang@daum.net
장희정

BAN	총점
2	265
3	210
1	170

```
SELECT BAN , SUM(KOR) 총점 FROM REPORT
GROUP BY  BAN  WITH ROLLUP; -- 전체소계
```



BAN	총점
1	170
2	265
3	210
NULL	645

순위함수

- ROW_NUMBER()
- RANK()
- DENSE_RANK()

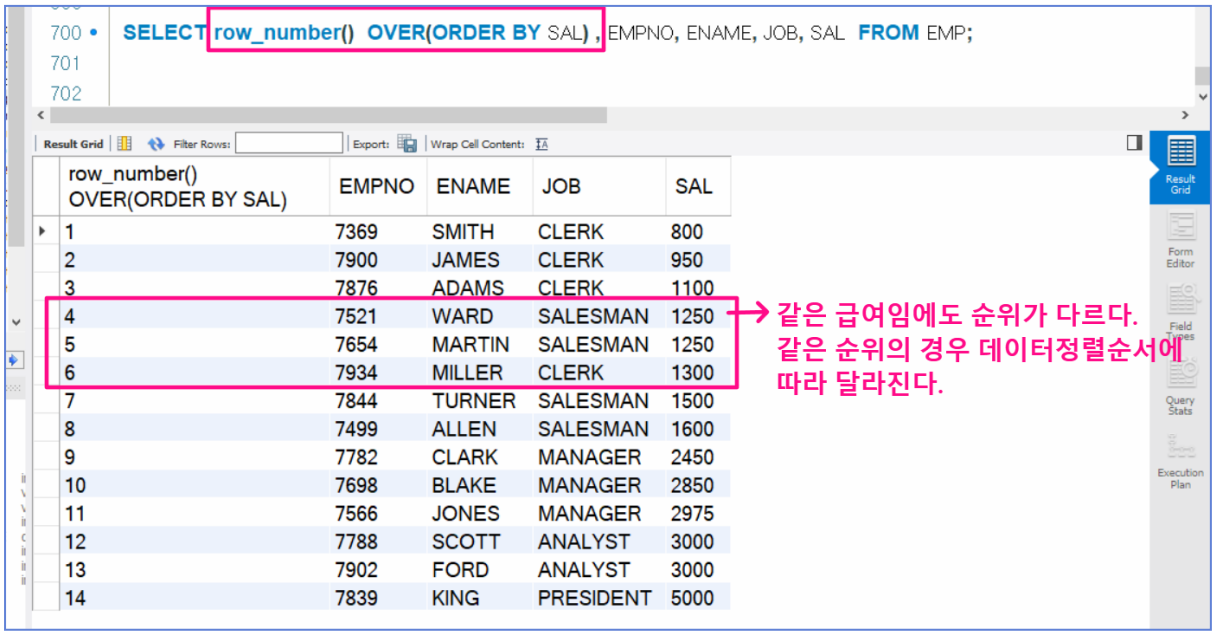
```
ROW_NUMBER() OVER ([PARTITION BY 열] ORDER BY 열)

RANK () OVER ([PARTITION BY 열] ORDER BY 열)

DENSE_RANK () OVER ([PARTITION BY 열] ORDER BY 열)
```

- 순위를 부여하기 위한 정렬 조건으로 OVER(ORDER BY 열)를 명시하여 오름차순 또는 내림차순으로 해당 열 데이터의 순위를 부여한다.
데이터 그룹별 순위를 부여하고 싶다면 PARTITION BY열 옵션을 사용한다.

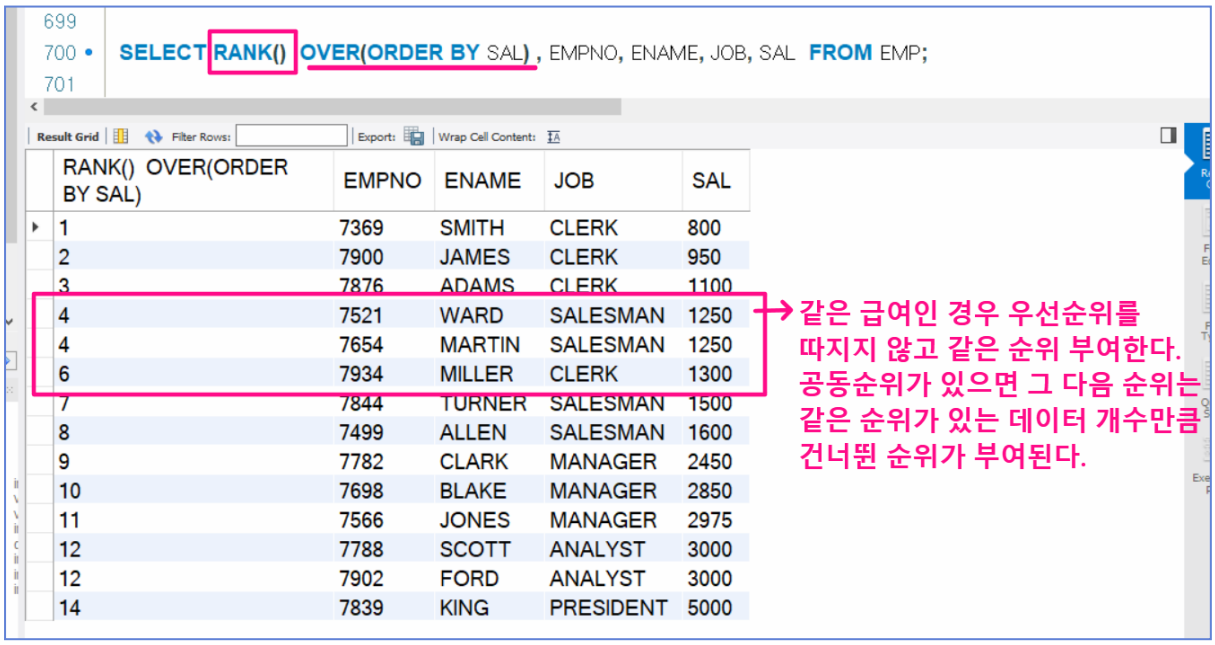
8253jang@daum.net
장희정



```
700 • SELECT row_number() OVER(ORDER BY SAL) , EMPNO, ENAME, JOB, SAL FROM EMP;
701
702
```

row_number() OVER(ORDER BY SAL)	EMPNO	ENAME	JOB	SAL
1	7369	SMITH	CLERK	800
2	7900	JAMES	CLERK	950
3	7876	ADAMS	CLERK	1100
4	7521	WARD	SALESMAN	1250
5	7654	MARTIN	SALESMAN	1250
6	7934	MILLER	CLERK	1300
7	7844	TURNER	SALESMAN	1500
8	7499	ALLEN	SALESMAN	1600
9	7782	CLARK	MANAGER	2450
10	7698	BLAKE	MANAGER	2850
11	7566	JONES	MANAGER	2975
12	7788	SCOTT	ANALYST	3000
13	7902	FORD	ANALYST	3000
14	7839	KING	PRESIDENT	5000

같은 급여임에도 순위가 다르다.
같은 순위의 경우 데이터정렬순서에 따라 달라진다.



```
699
700 • SELECT RANK() OVER(ORDER BY SAL) , EMPNO, ENAME, JOB, SAL FROM EMP;
701
```

RANK() OVER(ORDER BY SAL)	EMPNO	ENAME	JOB	SAL
1	7369	SMITH	CLERK	800
2	7900	JAMES	CLERK	950
3	7876	ADAMS	CLERK	1100
4	7521	WARD	SALESMAN	1250
4	7654	MARTIN	SALESMAN	1250
6	7934	MILLER	CLERK	1300
7	7844	TURNER	SALESMAN	1500
8	7499	ALLEN	SALESMAN	1600
9	7782	CLARK	MANAGER	2450
10	7698	BLAKE	MANAGER	2850
11	7566	JONES	MANAGER	2975
12	7788	SCOTT	ANALYST	3000
12	7902	FORD	ANALYST	3000
14	7839	KING	PRESIDENT	5000

같은 급여인 경우 우선순위를 따지지 않고 같은 순위 부여한다.
공동순위가 있으면 그 다음 순위는 같은 순위가 있는 데이터 개수만큼 건너뛴 순위가 부여된다.

8253jang@daum.net
장희정

```
700 • SELECT DENSE_RANK() OVER(ORDER BY SAL) , EMPNO, ENAME, JOB, SAL  FROM EMP;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

DENSE_RANK() OVER(ORDER BY SAL)	EMPNO	ENAME	JOB	SAL
1	7369	SMITH	CLERK	800
2	7900	JAMES	CLERK	950
3	7876	ADAMS	CLERK	1100
4	7521	WARD	SALESMAN	1250
4	7654	MARTIN	SALESMAN	1250
5	7934	MILLER	CLERK	1300
6	7844	TURNER	SALESMAN	1500
7	7499	ALLEN	SALESMAN	1600
8	7782	CLARK	MANAGER	2450
9	7698	BLAKE	MANAGER	2850
10	7566	JONES	MANAGER	2975
11	7788	SCOTT	ANALYST	3000
11	7902	FORD	ANALYST	3000
12	7839	KING	PRESIDENT	5000

→ 건너뛰지 않고 순위를 부여한다. 4위가 2개여도 그 다음 데이터는 5위가 된다.

```
712 • SELECT ROW_NUMBER() OVER( partition by DEPTNO ORDER BY JOB DESC) , DEPTNO , JOB , COUNT(*), MAX(SAL)
713   FROM EMP
714   group by DEPTNO , JOB;
715
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ROW_NUMBER() OVER(partition by DEPTNO ORDER BY JOB DESC)	DEPTNO	JOB	COUNT(*)	MAX(SAL)
1	10	PRESIDENT	1	5000
2	10	MANAGER	1	2450
3	10	CLERK	1	1300
1	20	MANAGER	1	2975
2	20	CLERK	2	1100
3	20	ANALYST	2	3000
1	30	SALESMAN	4	1600
2	30	MANAGER	1	2850
3	30	CLERK	1	950

MySQL에서 ROW_NUMBER()를 사용하여 특정 범위의 레코드 조회

WHERE 절에서는 별칭(alias)을 직접 사용할 수 없다.

```
704 • SELECT row_number() OVER(ORDER BY SAL) as no , EMPNO, ENAME, JOB, SAL
705   FROM EMP
706   where no <= 3;
707
```

Action Output

#	Time	Action	Message
1	19:04:36	SELECT row_number() OVER(ORDER BY SAL) as no , EMPNO, ENAME, JOB, SAL FROM EMP where ...	Error Code: 1054. Unknown column 'no' in 'where clause'

8253jang@daum.net
장희정

서브쿼리를 활용한다.

```
SELECT *
FROM (
    SELECT ROW_NUMBER() OVER (ORDER BY SAL) AS no,
            EMPNO, ENAME, JOB, SAL
    FROM EMP
) AS subquery
WHERE no BETWEEN 3 AND 6;
```

LIMIT을 활용한다.

```
SELECT EMPNO, ENAME, JOB, SAL
FROM EMP
ORDER BY SAL
LIMIT 4 OFFSET 2;
```

LIMIT 4 OFFSET 2 → 3번째부터 4개(3~6번) 가져옴
OFFSET 2 → 앞의 2개를 건너뛰고 시작
LIMIT 4 → 이후 4개(즉, 3~6번째) 가져오기