

20220804

🕒 작성일시	@2022년 8월 4일 오후 5:37
▼ 강의 번호	
📅 유형	
🔗 자료	
☑ 복습	<input type="checkbox"/>
☰ 속성	

Chapter 7 데이터 생성, 조작과 변환

7.1 문자열 데이터 처리

가변, 입력 길이에 따라 변환다.

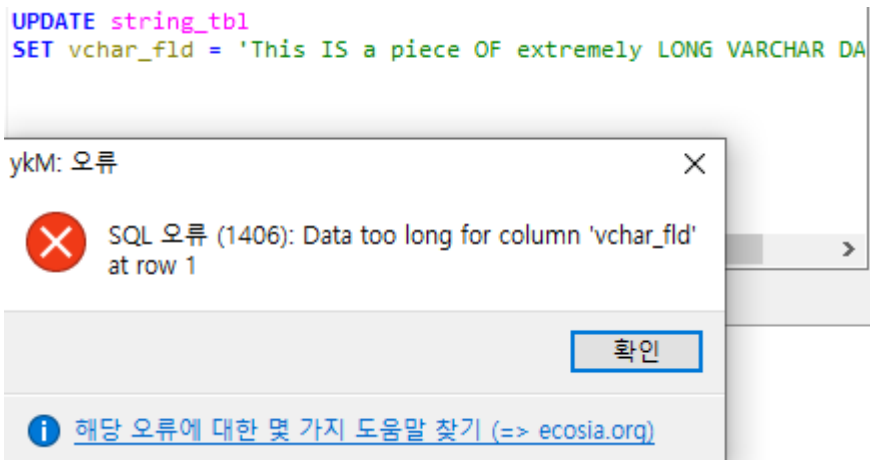
```
CREATE TABLE string_tbl
(char_fld CHAR(30),
vchar_fld VARCHAR(30),
text_fld TEXT
);
```

```
CREATE TABLE string_tbl
(char_fld CHAR(30),
vchar_fld VARCHAR(30),
text_fld TEXT
);
Query OK, 0 rows affected (0.02 sec)
```

7.1.1 문자열 생성

```
INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)
VALUES ('This is char data',
       'This is varchar data',
       'This is text data');
```

```
mysql> INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)
-> VALUES ('This is char data',
-> 'This is varchar data',
-> 'This is text data');
Query OK, 1 row affected (0.00 sec)
```



```
mysql> UPDATE string_tbl
-> SET vchar_fld = 'This is a piece of extremely long varchar data';
ERROR 1406 (22001): Data too long for column 'vchar_fld' at row 1
```

```
UPDATE string_tbl
SET vchar_fld = 'This IS a piece OF extremely LONG VARCHAR DATA';
```

```
mysql> SELECT vchar_fld
-> FROM string_tbl;
+-----+
| vchar_fld |
+-----+
| This is a piece of extremely l |
+-----+
1 row in set (0.05 sec)
```

```
SELECT vchar_fld
FROM string_tbl;
```

작은 따옴표(\') 포함(오라클 방식이지만 Mysql도 인식 됨)

```
mysql> UPDATE string_tbl
-> SET text_fld = 'This string didn't work, but it does now';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

NOTE_ 오라클 데이터베이스 및 MySQL을 사용할 경우, 다음과 같이 바로 앞에 백슬래시 문자를 추가해서 작은 따옴표를 무시하도록 할 수도 있습니다.

```
UPDATE string_tbl SET text_fld =
'This string didn\'t work, but it does now'
```

text_fld
This string didn't work, but it does now

```
UPDATE string_tbl
SET text_fld = 'This string didn\'t work, but it does now';
```

```
1 SELECT TEXT_fld
2 FROM string_tbl;
```

string_tbl (1r x 1c)

TEXT_fld
This string didn't work, but it does now

```
SELECT QUOTE(text_fld)
FROM string_tbl;
```

```
1 SELECT QUOTE(text_fld)
2 FROM string_tbl;
```

string_tbl (1r x 1c)

QUOTE(text_fld)
'This string didn't work, but it does now'

특수 문자 포함(including special characters)

```

mysql> SELECT CHAR(128,129,130,131,132,133,134,135,136,137);
+-----+
| CHAR(128,129,130,131,132,133,134,135,136,137) |
+-----+
| Çüéääåçëë                                     |
+-----+
1 row in set (0.01 sec)

mysql> SELECT CHAR(138,139,140,141,142,143,144,145,146,147);
+-----+
| CHAR(138,139,140,141,142,143,144,145,146,147) |
+-----+
| èïììÄÅÉæŒô                                     |
+-----+
1 row in set (0.01 sec)

mysql> SELECT CHAR(148,149,150,151,152,153,154,155,156,157);
+-----+
| CHAR(148,149,150,151,152,153,154,155,156,157) |
+-----+
| öðòùÿŸÜø£ø                                     |
+-----+
1 row in set (0.00 sec)

mysql> SELECT CHAR(158,159,160,161,162,163,164,165);
+-----+
| CHAR(158,159,160,161,162,163,164,165) |
+-----+
| ×fáíóúñÑ                                       |
+-----+
1 row in set (0.01 sec)

```

설정 때문에 똑같이 나오지 않을 수도 있다고 하심.

1	SELECT CHAR(148,149,150,151,152,153,154,155,156,157);
결과 #1 (1r × 1c)	
	CHAR(148,149,150,151,152,153,154,155,156,157) 뽕뽕뽕뽕뽕뽕뽕뽕뽕

```
SELECT CHAR(148,149,150,151,152,153,154,155,156,157);
```

7.1.2 문자열 조작(String Manipulation)

테이블의 모든 데이터를 완전 삭제

```
mysql> DELETE FROM string_tbl;
Query OK, 1 row affected (0.02 sec)

mysql> INSERT INTO string_tbl (char_fld, varchar_fld, text_fld)
-> VALUES ('This string is 28 characters',
-> 'This string is 28 characters',
-> 'This string is 28 characters');
Query OK, 1 row affected (0.00 sec)
```

```
DELETE FROM string_tbl;
```

DROP은 열까지 완전 삭제하는 명령어.

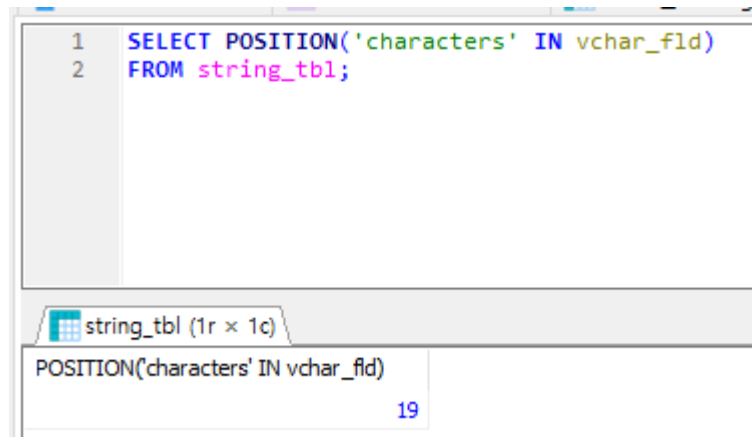
```
INSERT INTO string_tbl (char_fld, vchar_fld, text_fld)
VALUES('This string is 28 characters',
      'This string is 28 characters',
      'This string is 28 characters');
```

숫자를 반환하는 문자열 함수

```
1 SELECT LENGTH(char_fld) CHAR_LENGTH,
2    LENGTH(vchar_fld) varchar_length,
3    LENGTH(text_fld) text_length
4 FROM string_tbl;
```

string_tbl (1r × 3c)		
CHAR_LENGTH	varchar_length	text_length
28	28	28

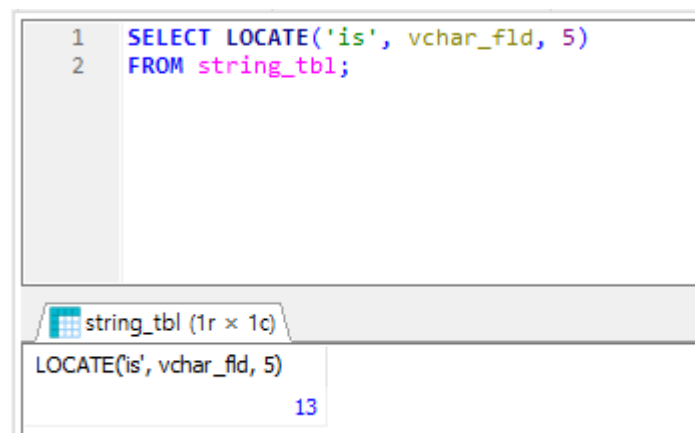
```
SELECT LENGTH(char_fld) CHAR_LENGTH,
       LENGTH(vchar_fld) varchar_length,
       LENGTH(text_fld) text_length
FROM string_tbl;
```



```
SELECT POSITION('characters' IN vchar_fld)
FROM string_tbl;
```

부분 문자열을 찾을 수 없는 경우 position() 함수는 0을 반환합니다.

vchar_fld 열의 다섯번째 문자에서 시작하는 문자열 'is'의 위치를 찾는 예제입니다.



```
SELECT LOCATE('is', vchar_fld, 5)
FROM string_tbl;
```

오라클에서는 position()과 locate()를 함수를 포함하지 않는다. (사용할 수 없다.)

```
DELETE FROM string_tbl;
=====
INSERT INTO string_tbl(vchar_fld)
```



```
VALUES ('abcd'),  
('xyz'),  
('QRSTU'),  
('qrstuv'),  
('12345');
```

```
mysql> DELETE FROM string_tbl;  
Query OK, 1 row affected (0.00 sec)  
  
mysql> INSERT INTO string_tbl(vchar_fld)  
-> VALUES ('abcd'),  
->          ('xyz'),  
->          ('QRSTU'),  
->          ('qrstuv'),  
->          ('12345');  
Query OK, 5 rows affected (0.05 sec)  
Records: 5  Duplicates: 0  Warnings: 0
```

다음은 정렬 순서에 따른 5개의 문자열입니다.

```
mysql> SELECT vchar_fld  
-> FROM string_tbl  
-> ORDER BY vchar_fld;
```

```
SELECT vchar_fld  
FROM string_tbl  
ORDER BY vchar_fld;
```

- 문자 비교

‘abcd’가 ‘xyz’ 작은 게 앞으로 -1

‘xyz’이 ‘qrstuv’ 큰 게 앞에 올 때는 1

대소문자는 구분하지 않으므로 ‘qrstuv’와 QRSTUV’는 같아서 ()

```
SELECT STRCMP('12345', '12345') 12345_12345,
       STRCMP('abcd', 'xyz') abcd_xyz,
       STRCMP('abcd', 'QRSTUV') QRSTUV,
       STRCMP('qrstuv', 'QRSTUV') qrstuv_QRSTUV,
       STRCMP('12345', 'xyz') 12345_xyz,
       STRCMP('xyz', 'qrstuv') xyz_qrstuv;
```

```
mysql> SELECT STRCMP('12345', '12345') 12345_12345,
       -> STRCMP('abcd', 'xyz') abcd_xyz,
       -> STRCMP('abcd', 'QRSTUV') abcd_QRSTUV,
       -> STRCMP('qrstuv', 'QRSTUV') qrstuv_QRSTUV,
       -> STRCMP('12345', 'xyz') 12345_xyz,
       -> STRCMP('xyz', 'qrstuv') xyz_qrstuv;
+-----+-----+-----+-----+-----+-----+
| 12345_12345 | abcd_xyz | abcd_QRSTUV | qrstuv_QRSTUV | 12345_xyz | xyz_qrstuv |
+-----+-----+-----+-----+-----+-----+
|          0 |        -1 |          -1 |             0 |        -1 |          1 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

LIKE 연산자

```
mysql> SELECT name, name LIKE '%y' ends_in_y
-> FROM category;
```

name	ends_in_y
Action	0
Animation	0
Children	0
Classics	0
Comedy	1
Documentary	1
Drama	0
Family	1
Foreign	0
Games	0
Horror	0
Music	0
New	0
Sci-Fi	0
Sports	0
Travel	0

16 rows in set (0.00 sec)

‘ns’로 끝나면 1을 반환하고 그렇지 않으면 0을 반환한다.

```

1  SELECT NAME, NAME LIKE '%ns' ends_in_y
2  FROM department;
```

NAME	ends_in_y
Operations	1
Loans	1
Administration	0

```
SELECT NAME, NAME LIKE '%ns' ends_in_y
FROM department;
```

다음과 같이 **regexp** 연산자를 사용할 수 있습니다.

```
mysql> SELECT name, name REGEXP 'y$' ends_in_y
-> FROM category;
```

name	ends_in_y
Action	0
Animation	0
Children	0
Classics	0
Comedy	1
Documentary	1

지우고 다시 삽입

```
mysql> DELETE FROM string_tbl;
Query OK, 5 rows affected (0.00 sec)

mysql> INSERT INTO string_tbl (text_fld)
-> VALUES ('This string was 29 characters');
Query OK, 1 row affected (0.01 sec)
```

```
DELETE FROM string_tbl;
```

```
INSERT INTO string_tbl(text_fld)
VALUES('this string was 29 characters');
```

```
UPDATE string_tbl  
SET text_fld = CONCAT(text_fld, ', but now it ts longer');
```

```
mysql> UPDATE string_tbl  
-> SET text_fld = CONCAT(text_fld, ', but now it is longer');  
Query OK, 1 row affected (0.03 sec)  
Rows matched: 1 Changed: 1 Warnings: 0
```

그 결과 수정된 `text_fld` 열의 내용은 다음과 같습니다.

```
mysql> SELECT text_fld  
-> FROM string_tbl;  
+-----+  
| text_fld |  
+-----+  
| This string was 29 characters, but now it is longer |  
+-----+  
1 row in set (0.00 sec)
```

<영문판>

호스트: 127.0.0.1 데이터베이스: bank 쿼리*	
1	SELECT CONCAT(fname, ' ', lname, ' has been a',
2	title, ' since ', start_date) emp_narrative
3	FROM employee
4	WHERE title = 'Teller' OR title = 'Head Teller';
employee (13r x 1c)	
emp_narrative	
Helen Fleming has been aHead Teller since 2004-03-17	
Chris Tucker has been aTeller since 2004-09-15	
Sarah Parker has been aTeller since 2002-12-02	
Jane Grossman has been aTeller since 2002-05-03	
Paula Roberts has been aHead Teller since 2002-07-27	
Thomas Ziegler has been aTeller since 2000-10-23	
Samantha Jameson has been aTeller since 2003-01-08	
John Blake has been aHead Teller since 2000-05-11	
Cindy Mason has been aTeller since 2002-08-09	
Frank Portman has been aTeller since 2003-04-01	
Theresa Markham has been aHead Teller since 2001-03-15	
Beth Fowler has been aTeller since 2002-06-29	
Rick Tulman has been aTeller since 2002-12-12	

<한글판>

```
SELECT CONCAT(first_name, ' ', last_name,
' has been a customer since ', DATE(create_date)) cust_narrative
FROM customer;
```

```
mysql> SELECT concat(first_name, ' ', last_name,
-> ' has been a customer since ', date(create_date)) cust_narrative
-> FROM customer;
```

cust_narrative
MARY SMITH has been a customer since 2006-02-14
PATRICIA JOHNSON has been a customer since 2006-02-14
LINDA WILLIAMS has been a customer since 2006-02-14
BARBARA JONES has been a customer since 2006-02-14
ELIZABETH BROWN has been a customer since 2006-02-14
JENNIFER DAVIS has been a customer since 2006-02-14
MARIA MILLER has been a customer since 2006-02-14
SUSAN WILSON has been a customer since 2006-02-14

문자열을 반환하는 concat()함수는 문자열이 저장된 데이터를 바꾸기 위한 용도로 사용할 수 있고,

UPDATE - 영구 변환

각 데이터 조각을 합쳐서 문자열을 만들어 임시로 보는 용도로도 사용된다.

SELECT - 임시 변환

```
SELECT first_name || ' ' || last_name ||  
       ' has been a customer since ' || date(create_date)) cust_narrative  
FROM customer;
```

SQL 서버에는 `concat()` 함수가 포함되지 않으므로 이전 쿼리와 동일하게 처리하되 `||` 대신 SQL 서버의 연결 연산자(+)를 사용해야 합니다.

`concat()`은 문자열의 시작 또는 끝에 문자를 추가할 때 유용하지만, 문자열 **중간**에 문자를 추가하거나 해당 문자를 교체해야 할 때도 쓸 수 있습니다. 세 종류의 데이터베이스 서버 모두 이 기능을 제공하지만 서로 다르므로, MySQL 서버에서의 기능을 살펴본 뒤 다른 두 서버의 기능을 보여드리겠습니다.

MySQL에는 '원래 문자열, 시작 위치, 대체할 문자 개수, 대체 문자열'의 네 개 인수를 가지는 `insert()` 함수가 포함됩니다. 세 번째 인수의 값에 따라 문자열에 문자를 삽입하거나 바꿀 수 있습니다. 세 번째 인수의 값이 0이면 대체 문자열이 삽입되며 다음과 같이 처리됩니다.

나누기의 나머지


```
mysql> SELECT MOD(10,4);
+-----+
| MOD(10,4) |
+-----+
|          2 |
+-----+
1 row in set (0.02 sec)
```

mod() 함수의 인수로는 보통 정수를
도 있습니다.

```
mysql> SELECT MOD(22.75, 5);
+-----+
| MOD(22.75, 5) |
+-----+
|          2.75 |
+-----+
1 row in set (0.02 sec)
```

2의 8승

1	SELECT POW(2, 8);
결과 #1 (1r × 1c)	
POW(2, 8)	256

7.2.2 숫자 자릿수 관리

1	SELECT CEIL(72.445), FLOOR(72.445);
결과 #1 (1r × 2c)	
CEIL(72.445)	FLOOR(72.445)
73	72

ceil()은 작은 부분도 반올림

round()함수에서 두번째 인수 즉 반올림의 위치값을 생략하면 자동으로 정수를 기준으로 자동 반올림 처리한다.

소수 부분의 숫자가 두 정수 사이의 중간 또는 그 이상일 때는 반올림되지만, 두 정수 사이의 중간보다 작으면 내림됩니다.

호스트: 127.0.0.1 | 데이터베이스: bank | 테이블: string_tbl | 데이터 | 쿼리

```
1 SELECT ROUND(72.49999), ROUND(72.5), ROUND(72.50001);
```

결과 #1 (1r × 3c)

ROUND(72.49999)	ROUND(72.5)	ROUND(72.50001)
72	73	73

round() 함수의 두 번째 인수인 위치값이 함수일때는 소수점 아래의 ~로 반올림하고 음수일 때는 소수점 위의 ~에서 반올림한다.

호스트: 127.0.0.1 | 데이터베이스: bank | 테이블: string_tbl | 데이터 | 쿼리

```
1 SELECT ROUND(72.0909, 1), ROUND(72.0909, 2), ROUND(72.0909, 3);
```

결과 #1 (1r × 3c)

ROUND(72.0909, 1)	ROUND(72.0909, 2)	ROUND(72.0909, 3)
72.1	72.09	72.091

```
SELECT ROUND(72.0909, 1), ROUND(72.0909, 2), ROUND(72.0909, 3);
```

`round()` 함수와 마찬가지로 `truncate()` 함수도 두 번째 인수를 이용해서 소수점 오른쪽 자릿수를 지정하도록 허용하지만, `truncate()` 함수는 반올림하는 대신 원치 않는 숫자를 버립니다. 다음 예제는 `truncate()` 함수를 이용해서 숫자 72.0909를 소수점 1, 2, 3자리에 맞춰 자르는 방법을 보여줍니다.

```
mysql> SELECT TRUNCATE(72.0909, 1), TRUNCATE(72.0909, 2),
-> TRUNCATE(72.0909, 3);
```

TRUNCATE(72.0909, 1)	TRUNCATE(72.0909, 2)	TRUNCATE(72.0909, 3)
72.0	72.09	72.090

1 row in set (0.00 sec)

NOTE_ SQL 서버에는 `truncate()` 함수가 없습니다. 대신 `round()` 함수는 선택적인 세 번째 인수를 허용하므로, 이 값이 0이 아닐 경우에는 숫자가 반올림되지 않고 잘려나갑니다.

```
1 SELECT ROUND(17, -1), TRUNCATE(17, -1);
```

결과 #1 (1r x 2c)	
ROUND(17, -1)	TRUNCATE(17, -1)
20	10

```
mysql> SELECT ROUND(17, -1), TRUNCATE(17, -1);
+-----+-----+
| ROUND(17, -1) | TRUNCATE(17, -1) |
+-----+-----+
|          20 |          10 |
+-----+-----+
1 row in set (0.00 sec)
```

7.3 시간 데이터 처리, 작업(Working with Temporal Data)

GMT - 영국 그리니치 표준시

UTC - 협정 세계 표준시 coordinated universal time

7.3.2 시간 데이터 생성 Generating

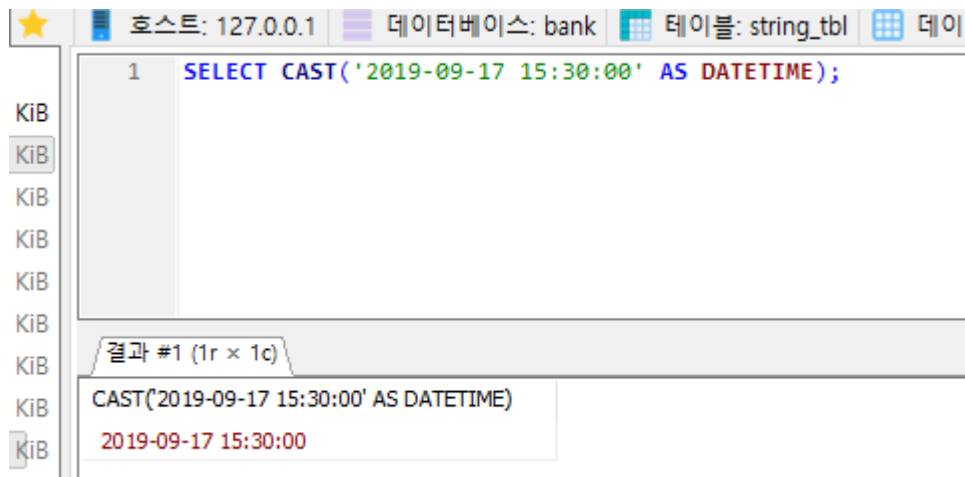
문자열을 날짜로 변환

CAST()

문자열을 날짜로 변환

서버가 `datetime` 값이라고 판단할 수 없거나 기본 형식이 아닌 형식을 사용해서 `datetime`을 표시하려는 경우, 서버는 문자열을 `datetime`으로 변환해야 합니다. 예를 들어 다음은 `cast()` 함수를 써서 `datetime` 값을 반환하는 간단한 쿼리입니다.

```
mysql> SELECT CAST('2019-09-17 15:30:00' AS DATETIME);
+-----+
| CAST('2019-09-17 15:30:00' AS DATETIME) |
+-----+
| 2019-09-17 15:30:00                      |
+-----+
1 row in set (0.00 sec)
```

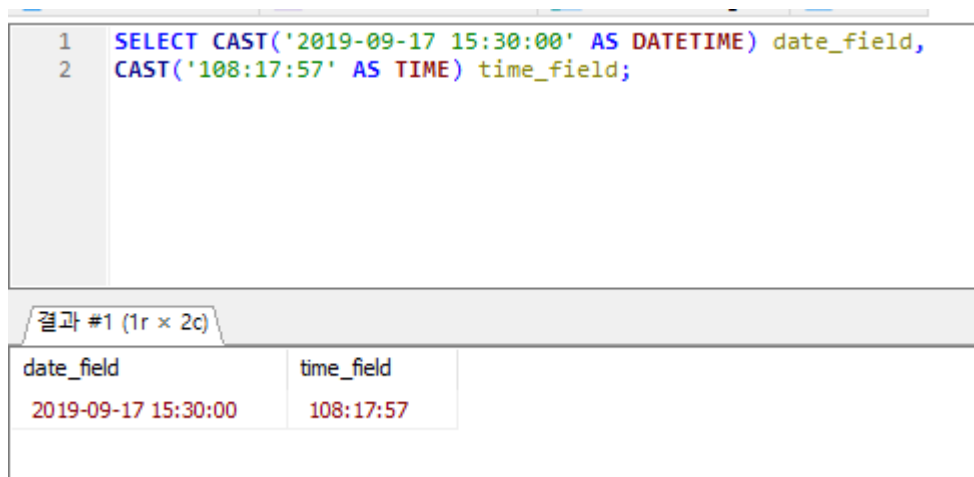


The screenshot shows a database client window with the following details:

- Host: 127.0.0.1
- Database: bank
- Table: string_tbl
- Query: `SELECT CAST('2019-09-17 15:30:00' AS DATETIME);`
- Result: A single row containing the value `2019-09-17 15:30:00`.

결과 #1 (1r x 1c)	
	CAST('2019-09-17 15:30:00' AS DATETIME)
	2019-09-17 15:30:00

```
SELECT CAST('2019-09-17 15:30:00' AS DATETIME);
```



The screenshot shows a database client window with the following details:

- Query: `SELECT CAST('2019-09-17 15:30:00' AS DATETIME) date_field, CAST('108:17:57' AS TIME) time_field;`
- Result: A single row with two columns: `date_field` (2019-09-17 15:30:00) and `time_field` (108:17:57).

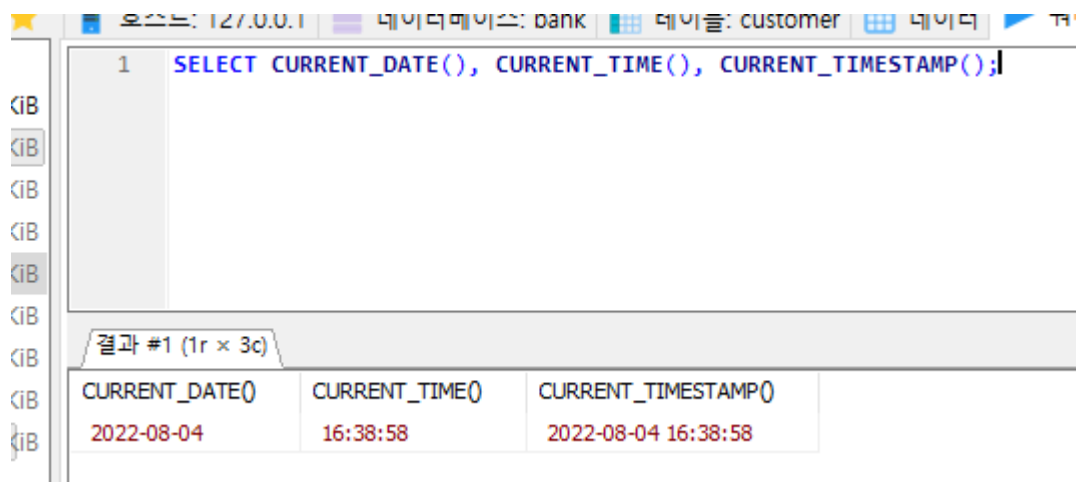
결과 #1 (1r x 2c)	
date_field	time_field
2019-09-17 15:30:00	108:17:57

```
SELECT CAST('2019-09-17 15:30:00' AS  
CAST('108:17:57' AS TIME) time_field;
```

날짜 생성 관련 함수

```
UPDATE rental  
SET return_date = STR_TO_DATE('September 17, 2019', '%M %d, %y')  
WHERE rental_id = 99999;
```

```
SELECT CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP();
```



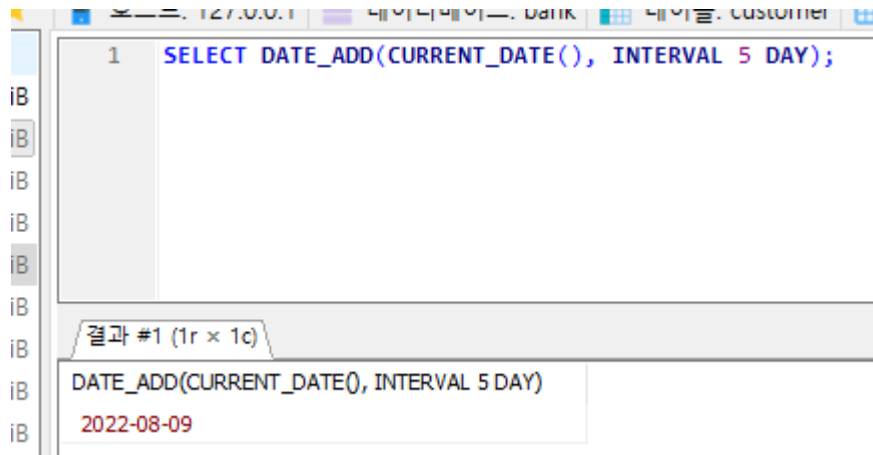
The screenshot shows a SQL client window with the following details:

- Host: 127.0.0.1
- Database: bank
- Table: customer
- Query: `SELECT CURRENT_DATE(), CURRENT_TIME(), CURRENT_TIMESTAMP();`
- Result set: 결과 #1 (1r x 3c)
- Columns: `CURRENT_DATE()`, `CURRENT_TIME()`, `CURRENT_TIMESTAMP()`
- Values: `2022-08-04`, `16:38:58`, `2022-08-04 16:38:58`

CURRENT_DATE()	CURRENT_TIME()	CURRENT_TIMESTAMP()
2022-08-04	16:38:58	2022-08-04 16:38:58

7.3.3 시간데이터 조작

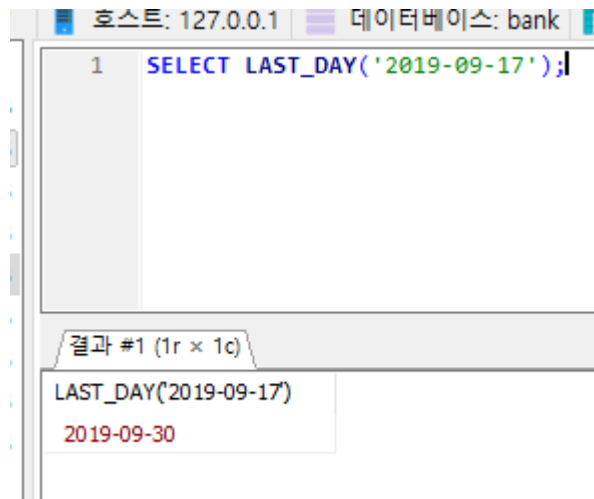
날짜를 반환하는 시간 함수



```
SELECT DATE_ADD(CURRENT_DATE(), INTERVAL 5 DAY);
```

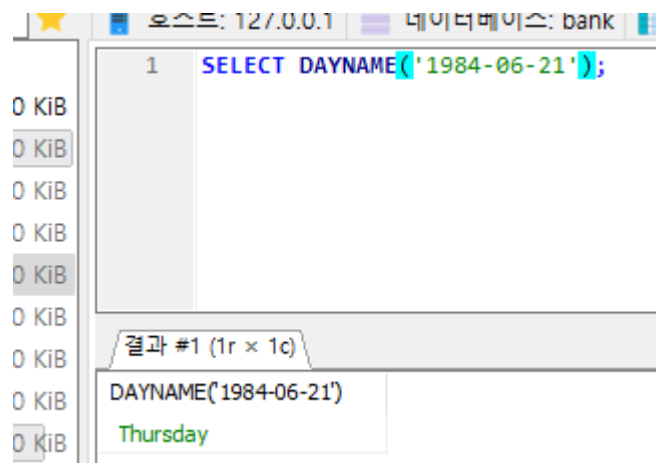
만약 고객이 2019년 9월 17일에 이체를 요청할 경우 다음 예제에서처럼 9월의 마지막 날을 찾을 수 있습니다.

```
mysql> SELECT LAST_DAY('2019-09-17');
+-----+
| LAST_DAY('2019-09-17') |
+-----+
| 2019-09-30              |
+-----+
1 row in set (0.10 sec)
```

```
SELECT LAST_DAY('2019-09-17');
```

문자열을 반환하는 시간 함수 Temporal function that retrun strings



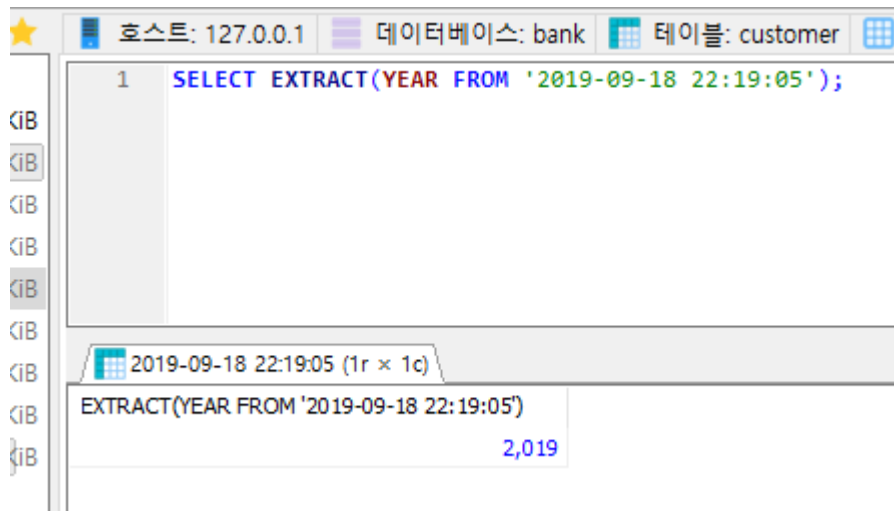
문자열을 반환하는 시간 함수

문자열 값을 반환하는 대부분의 시간 함수는 다. 예를 들어 MySQL에는 다음과 같이 특 함수가 포함됩니다.

```
mysql> SELECT DAYNAME('2019-09-18');
+-----+
| DAYNAME('2019-09-18') |
+-----+
| Wednesday              |
+-----+
1 row in set (0.00 sec)
```

`extract()` 함수는 `date_add()` 함수와 동일한 기간 자료형 ([표 7-5] 참조)으로 원하는 날짜 요소를 정의합니다. 예를 들어 `datetime` 값에서 연도 부분만 추출하려면 다음과 같이 수행합니다.

```
mysql> SELECT EXTRACT(YEAR FROM '2019-09-18 22:19:05');
+-----+
| EXTRACT(YEAR FROM '2019-09-18 22:19:05') |
+-----+
| 2019 |
+-----+
1 row in set (0.00 sec)
```



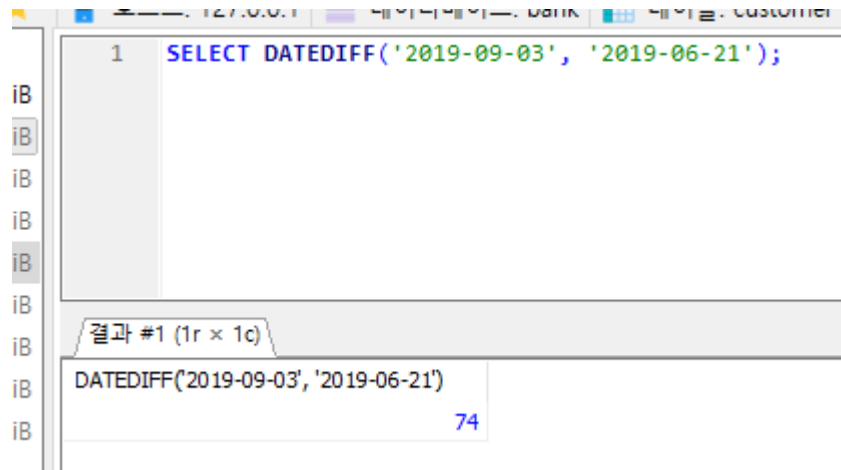
숫자를 반환하는 시간 함수 Temporal function that return numbers

숫자를 반환하는 시간 함수

7장 앞부분에서 주어진 기간을 날짜 값에 더하여 다른 날짜 값 생성에 사용하는 함수를 설명했습니다. 날짜로 작업할 때의 또 다른 일반적인 작업은 **두 개의 날짜 값을 가져와 두 날짜 사이의 기간(년, 주, 일)을 계산하는 것**입니다. MySQL에는 두 날짜 사이의 전체 일 수를 반환하는 `datediff()` 함수가 포함되어 있습니다.

예를 들어 이번 여름에 아이들이 등교하지 않는 기간을 알고 싶다면 다음과 같이 수행합니다.

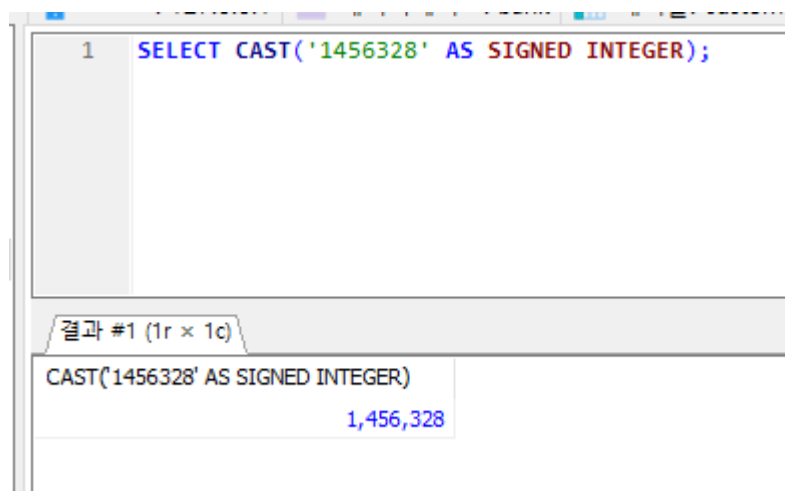
```
mysql> SELECT DATEDIFF('2019-09-03', '2019-06-21');
+-----+
| DATEDIFF('2019-09-03', '2019-06-21') |
+-----+
| 74 |
+-----+
1 row in set (0.00 sec)
```



7.4 Conversion Functions 변환 함수

cast() 함수를 사용하려면 값 또는 표현식, as 키워드 및 변환할 값의 자료형을 제공해야 합니다. 다음은 문자열을 정수로 변환하는 예제입니다.

CASE() 형변환



문자열을 숫자로 변환할 때는 cast()함수는 전체 문자열을 왼쪽에서 오른쪽으로 변환을 시도한다. 문자열에 숫자가 아닌 문자가 있으면 오류 없이 변환이 중지됩니다. 다음 예제를 살펴봅시다.

```
mysql> SELECT CAST('999ABC111' AS UNSIGNED INTEGER);
```

CAST('999ABC111' AS UNSIGNED INTEGER)
999

1 row in set, 1 warning (0.08 sec)

```
mysql> show warnings;
```

Level	Code	Message
Warning	1292	Truncated incorrect INTEGER value: '999ABC111'

1 row in set (0.07 sec)

이 경우 문자열의 처음 세 자리만 변환되고 나머지 문자열은 버려져 값은 999가 됩니다. 그러나 서버는 모든 문자열이 변환되지 않았음을 알리는 경고를 발생합니다.

문자열을 date, time, 또는 datetime 값으로 변환할 경우 형식 문자열과 함께 cast() 함수를 사용할 수 없으므로 각 자료형의 기본 형식을 사용해야 합니다. 날짜 문자열이 기본 형식이 아닐 경우(예를 들면 datetime 자료형일 때 YYYY-MM-DD HH:MI:SS), 이전 장에서 설명한 MySQL의 str_to_date() 함수와 같은 다른 함수를 사용합니다.

1	/*SELECT CAST('999ABC111' AS UNSIGNED INTEGER);*/	
2		
3	SHOW WARNINGS;	

결과 #1 (1r × 3c)		
Level	Code	Message
Warning	1,292	Truncated incorrect INTEGER value: '999ABC111'

이 경우 문자열의 처음 세 자리만 변환되고 나머지 문자열은 버려져 값은 999가 됩니다. 모든 문자열이 변환되지 않았음을 알리는 경고를 발생합니다.

Chapter 8 그룹화와 집계

8.1 그룹화의 개념

호스트: 127.0.0.1 데이터베이스: ban	
1	SELECT open_emp_id
2	FROM account
3	GROUP BY open_emp_id;

account (4r × 1c)	
open_emp_id	
1	
10	
13	
16	

```

1 SELECT open_emp_id, COUNT(*) how_many
2 FROM account
3 GROUP BY open_emp_id;

```

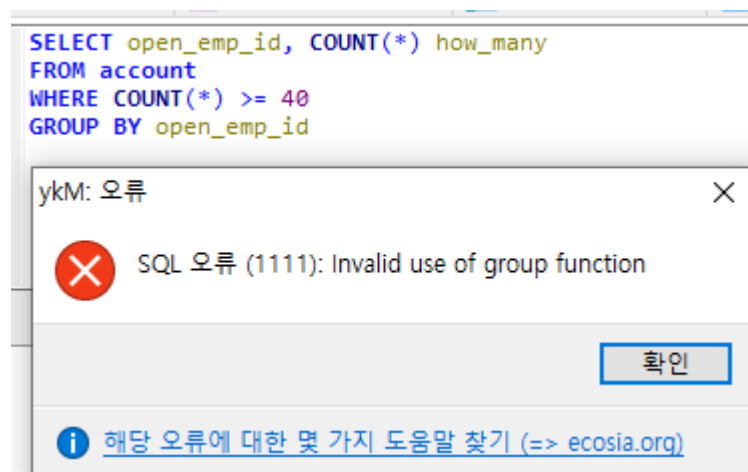
open_emp_id	how_many
1	8
10	7
13	3
16	6

1번 id 사람이 8개

10번 id 사람이 7개

.

.




WHERE 하고 나서 그룹화는 안 되서, HAVING 쓴다고 함.

호스트: 127.0.0.1 데이터베이스: bank 테이블: customer

```
1 SELECT open_emp_id, COUNT(*) how_many
2 FROM account
3 GROUP BY open_emp_id
4 HAVING COUNT(*) >= 4;
```

account (3r × 2c)

open_emp_id		how_many
1		8
10		7
16		6

- 번역 숙제

이 쿼리의 결과는 계좌 테이블 안에 10개 계좌 체크를 통해 당신에게 말한다. 거기에는 최대 잔고가 \$38,552.05이다. 가장 적은 잔고는 \$122.37 이다. 평균 잔고는 7,300.80 이다. 그리고 10개 계좌의 총 잔고는 \$73,008.01 이다.