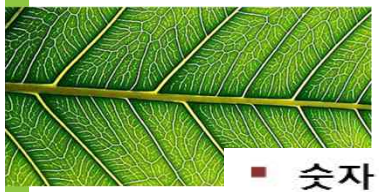
제4장
데이터 형식과 내장함수-1

1. 데이터 형식의 종류

- 숫자 데이터 형식
  - DECIMAL 형식은 정확한 수치를 저장하고 FLOAT, REAL 형식은 근사치를 저장
  - 소수점이 있는 실수는 되도록 DECIMAL 형식을 사용하여 저장하는 것이 바람직
  - 예를 들어 -999999.99~999999.99 범위의 숫자를 저장할 때는 DECIMAL(9,2)로 설정
  - 어떤 숫자를 부호 없는 정수로 지정하면
    - TINYINT는 0~255, SMALLINT는 0~65535,
    - MEDIUMINT는 0~16777215,
    - INT는 0~약 42억,
    - BIGINT는 0~약 1800경으로 표현할 수 있음
  - 부호 없는 정수를 지정할 때는 뒤에 UNSIGNED 예약어를 붙임



1. 데이터 형식의 종류

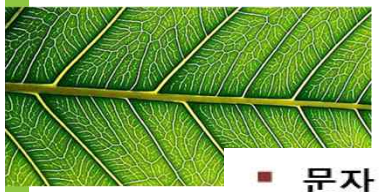
- 숫자 데이터 형식

표 7-1 숫자 데이터 형식의 종류

데이터 형식	바이트 수	숫자 범위	설명
BIT(N)	N/8		• 1~64bit 표현 • b'0000' 형식으로 저장
TINYINT	1	-128~127	• 정수 저장
BOOL BOOLEAN	1	-128~127	• 정수 저장 • TINYINT(1)과 동일 • 0은 false로, 그 외는 true로 취급
SMALLINT	2	-32768~32767	• 정수 저장
MEDIUMINT	3	-8388608~8388607	• 정수 저장
INT INTEGER	4	약 -21억~21억	• 정수 저장
BIGINT	8	약 -900경~900경	• 정수 저장
FLOAT	4	-3.40E+38~-1.17E-38	• 소수점 이하 7자리까지 저장
DOUBLE REAL	8	-1.22E-308~1.79E+308	• 소수점 이하 15자리까지 저장
DECIMAL(m,[d]) DEC(m,[d]) FIXED(m,[d]) NUMERIC(m,[d])	5~17	-1038+1~1038-1	• 전체 자릿수(m)와 소수점 이하 자릿수(d)를 가진 숫자 저장 예: DECIMAL(5,2)는 전체 자릿수를 5자리로 하되, 그중 소수점 이하를 2자리로 하겠다는 뜻

1. 데이터 형식의 종류

- 문자 데이터 형식
  - CHAR 형식은 고정 길이 문자형을 저장하고 자릿수가 고정되어 있음
  - VARCHAR 형식은 가변 길이 문자형을 저장
  - BINARY와 VARBINARY 형식은 바이트 단위의 이진 데이터 값을 저장
  - TEXT 형식은 대용량 글자를 저장하기 위한 형식으로, 필요한 크기에 따라서 TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT 등의 형식을 사용할 수 있음
  - BLOB(Binary Large OBject) 형식은 사진, 동영상, 문서 파일 등의 대용량 이진 데이터를 저장
  - ENUM 형식은 열거형 데이터를 저장하는 데 사용
  - SET 형식은 최대 64개의 데이터를 2개씩 세트로 묶어서 저장할 때 사용



1. 데이터 형식의 종류

- 문자 데이터 형식

표 7-2 문자 데이터 형식의 종류

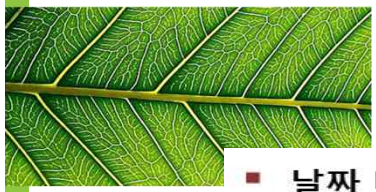
데이터 형식		바이트 수	설명
CHAR(n)		1~255	• 고정 길이 문자형 저장(character의 약자) • n을 1~255까지 지정 • CHAR만 쓰면 CHAR(1)과 동일
VARCHAR(n)		1~65535	• 가변 길이 문자형 저장(variable character의 약자) • n을 1~65535까지 지정
BINARY(n)		1~255	• 고정 길이의 이진 데이터 값 저장
VARBINARY(n)		1~255	• 가변 길이의 이진 데이터 값 저장
TEXT 형식	TINYTEXT	1~255	• 255 크기의 TEXT 데이터 값 저장
	TEXT	1~65535	• N 크기의 TEXT 데이터 값 저장
	MEDIUMTEXT	1~16777215	• 16777215 크기의 TEXT 데이터 값 저장
	LONGTEXT	1~4294967295	• 최대 4GB 크기의 TEXT 데이터 값 저장
BLOB 형식	TINYBLOB	1~255	• 255 크기의 BLOB 데이터 값 저장
	BLOB	1~65535	• N 크기의 BLOB 데이터 값 저장
	MEDIUMBLOB	1~16777215	• 16777215 크기의 BLOB 데이터 값 저장
	LONGBLOB	1~4294967295	• 최대 4GB 크기의 BLOB 데이터 값 저장
ENUM(값들 …)		1 또는 2	• 최대 65535개의 열거형 데이터 값 저장
SET(값들 …)		1, 2, 3, 4, 8	• 최대 64개의 서로 다른 데이터 값 저장

1. 데이터 형식의 종류

- 날짜와 시간 데이터 형식

표 7-3 날짜와 시간 데이터 형식의 종류

데이터 형식	바이트 수	설명
DATE	3	• 'YYYY-MM-DD' 형식으로 날짜 저장 • 저장 범위는 1001-01-01~9999-12-31
TIME	3	• 'HH:MM:SS' 형식으로 시간 저장 • 저장 범위는 -838:59:59.000000~838:59:59.000000
DATETIME	8	• 'YYYY-MM-DD HH:MM:SS' 형식으로 날짜와 시간 저장 • 저장 범위는 1001-01-01 00:00:00~9999-12-31 23:59:59
TIMESTAMP	4	• 'YYYY-MM-DD HH:MM:SS' 형식으로 날짜와 시간 저장 • 저장 범위는 1001-01-01 00:00:00~9999-12-31 23:59:59 • time_zone 시스템 변수와 관련이 있으며 UTC 시간대로 변환하여 저장
YEAR	1	• 'YYYY' 형식으로 연도 저장 • 저장 범위는 1901~2155



1. 데이터 형식의 종류

- 날짜 데이터 형식과 시간 데이터 형식의 차이

```
SELECT CAST('2020-10-19 12:35:29.123' AS DATE) AS 'DATE';
SELECT CAST('2020-10-19 12:35:29.123' AS TIME) AS 'TIME';
SELECT CAST('2020-10-19 12:35:29.123' AS DATETIME) AS 'DATETIME';
```

	Variable_name	Value
▶	character_set_system	utf8

- 기타 데이터 형식

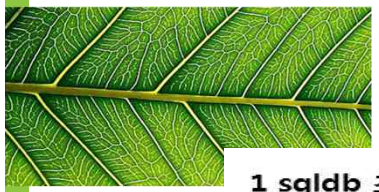
표 7-4 기타 데이터 형식의 종류

데이터 형식	바이트 수	설명
GEOMETRY POINT LINESTRING POLYGON	N/A	• 공간 데이터를 저장하는 형식으로 선, 점, 다각형 같은 공간 데이터 개체를 저장하고 조작
JSON	8	• JSON(JavaScript Object Notation) 문서 저장

2. 변수

- 변수 사용 형식

```
SET @변수이름 = 변수값; -- 변수 선언 및 값 대입
SELECT @변수이름; -- 변수 값 출력
```



2. 변수(실습)

1 sqldb 초기화하기

1-1 sqldb 초기화

1-2 열린 쿼리 창 모두 닫고 새 쿼리 창 열기

2 변수 사용하기

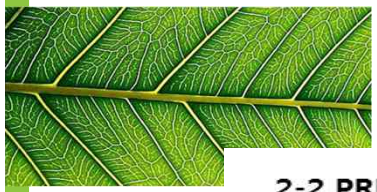
2-1 변수 선언하고 값 대입한 후 출력하기

```
USE sqldb;

SET @myVar1 = 5;
SET @myVar2 = 3;
SET @myVar3 = 4.25;
SET @myVar4 = 'MC 이름==> ';

SELECT @myVar1;
SELECT @myVar2 + @myVar3;

SELECT @myVar4 , userName FROM userTBL WHERE height > 180;
```

@myVar1
5

@myVar2 + @myVar3
7.250000000000000000000000000000

@myVar4	userName
MC 이름==>	강호동
MC 이름==>	박수홍



2. 변수(실습)

2-2 PREPARE 문과 EXECUTE 문에서 변수 활용

```
SET @myVar1 = 3;
PREPARE myQuery
    FROM 'SELECT userName, height FROM userTBL ORDER BY height LIMIT ?';
EXECUTE myQuery USING @myVar1;
```

	userName	height
	이경규	170
	김국진	171
	김제동	173

3. 데이터 형식 변환 함수

- 데이터 형식 변환 함수
  - 일반적으로 사용되는 데이터 형식 변환 함수는 CAST( )와 CONVERT( )
  - 두 함수는 기능이 거의 비슷

```
CAST(expression AS 데이터형식 [(길이)])
CONVERT(expression, 데이터형식 [(길이)])
```

- sqldb의 구매 테이블 (buyTBL)에서 평균 구매 개수를 구하는 쿼리문

```
USE sqldb;
SELECT AVG(amount) AS '평균 구매 개수' FROM buyTBL;
```

	평균 구매 개수
▶	2.9167

- 구매 개수를 정수로 출력

```
SELECT CAST(AVG(amount) AS SIGNED INTEGER) AS '평균 구매 개수' FROM buyTBL;
또는
SELECT CONVERT(AVG(amount), SIGNED INTEGER) AS '평균 구매 개수' FROM buyTBL;
```

	평균 구매 개수
▶	3

3. 데이터 형식 변환 함수

- **CAST( ) 함수를 사용하면 다양한 구분자($, /, %, @)를 날짜 형식(-)으로도 변경할 수 있음**

```
SELECT CAST('2020$12$12' AS DATE);
SELECT CAST('2020/12/12' AS DATE);
SELECT CAST('2020%12%12' AS DATE);
SELECT CAST('2020@12@12' AS DATE);
```

	CAST('2020@12@12' AS DATE)
	2020-12-12

- **단가(price)와 수량(amount)을 곱한 실제 입금액을 출력하는 쿼리문**

```
SELECT num, CONCAT(CAST(price AS CHAR(10)), 'X', CAST(amount AS CHAR(4)), '=') AS '단가X수량
', price * amount AS '구매액'
   FROM buyTBL;
```

	num	단가X수량	구매액
▶	1	30X2=	60
	2	1000X1=	1000
	3	200X1=	200
	4	[illegible]X5=	1000
	[illegible]	50.	[illegible]
	10	30X2=	60
	11	15X1=	15
	12	30X2=	60

3. 데이터 형식 변환 함수

- 암시적인 형 변환
  - CAST( ) 함수나 CONVERT( ) 함수를 사용하지 않고 데이터 형식을 변환하는 것

- 암시적인 형 변환 예

```
1 SELECT '100' + '200'; -- 문자와 문자를 더함(정수로 변환한 후 처리)
2 SELECT CONCAT('100', '200'); -- 문자와 문자를 연결(문자열 그대로 처리)
3 SELECT CONCAT(100, '200'); -- 정수와 문자를 연결(정수를 문자로 변환하여 처리)
4 SELECT 1 > '3mega'; -- 정수인 3으로 변환한 후 비교
5 SELECT 4 > '3MEGA'; -- 정수인 3으로 변환한 후 비교
6 SELECT 0 = 'mega3'; -- 문자가 0으로 변환됨
```

'100' + '200'	CONCAT('100', '200')	CONCAT(100, '200')	1 > '3mega'	4 > '3MEGA'	0 = 'mega3'
300	100200	100200	0	1	1

  - 1행 : 더하기 연산이므로 문자열을 정수로 변환한 후 처리
  - 2행 : CONCAT( )은 문자열을 연결하는 함수이므로 문자열 그대로 처리
  - 3행 : CONCAT( ) 함수 안의 정수 100을 문자열로 변환한 후 처리
  - 4행 : 비교 연산으로 앞에 '3'이 들어간 문자열이 숫자 3으로 변경되어 결국 '1>3'으로 처리
  - 5행 : 4행과 같은 방식으로 처리
  - 6행 : 앞에 'm'이 들어간 문자열이 숫자로 변경되면 그냥 0이 되므로 결국 '0=0'으로 처리

4. 내장 함수(제어 함수)

- IF(수식, 참, 거짓)
  - 수식이 참이면 두 번째 인수를 반환하고, 거짓이면 세 번째 인수를 반환
  - 다음 쿼리문을 실행하면 '거짓이다'가 출력

```
SELECT IF(100>200, '참이다', '거짓이다');
```

- IFNULL(수식1, 수식2)
  - 수식1이 NULL이 아니면 수식1을 반환하고, 수식1이 NULL이면 수식2를 반환
  - 다음 쿼리문의첫 번째는 '널이군요'가 출력되고, 두 번째는 '100'이 출력

```
SELECT IFNULL(NULL, '널이군요'), IFNULL(100, '널이군요');
```

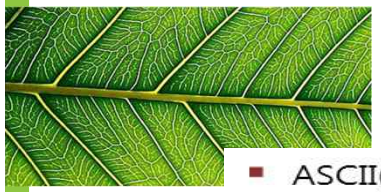
- NULLIF(수식1, 수식2)
  - 수식1과 수식2가 같으면 NULL을 반환하고, 다르면 수식1을 반환
  - 다음 쿼리문의 첫 번째는 ' NULL'이 출력되고, 두 번째는 '200'이 출력

```
SELECT NULLIF(100, 100), IFNULL(200, 100);
```

4. 내장 함수(제어 함수)

- CASE ... WHEN ... ELSE ... END
  - CASE는 내장 함수가 아니라 연산자(operator)
  - 다중 분기에 사용
  - 다음 예에서는 CASE 뒤의 값이 10이므로 세 번째 WHEN이 수행되어 '십'이 출력되고 해당하는 사항이 없다면 ELSE 부분이 출력됨

```
SELECT CASE 10
        WHEN 1 THEN '일'
        WHEN 5 THEN '오'
        WHEN 10 THEN '십'
        ELSE '모름'
END;
```



4. 내장 함수(문자열 함수)

- ASCII(아스키코드), CHAR(숫자)
  - 문자의 아스키코드 값을 반환하거나 숫자의 아스키코드 값에 해당하는 문자를 반환
  - 다음 쿼리 문의 첫 번째는 '65'가 출력되고, 두 번째는 'A'가 출력

```
SELECT ASCII('A'), CHAR(65);
```

- BIT_LENGTH(문자열), CHAR_LENGTH(문자열), LENGTH(문자열)
  - BIT_LENGTH( ) 함수는 할당된 비트 크기를 반환
  - CHAR_LENGTH( ) 함수는 문자의 개수를 반환
  - LENGTH( ) 함수는 할당된 바이트 수를 반환
  - MySQL은 기본적으로 UTF-8 코드를 사용하기 때문에 영문은 문자당 1바이트를, 한글은 문자당 3바이트를 할당

```
SELECT BIT_LENGTH('abc'), CHAR_LENGTH('abc'), LENGTH('abc');
SELECT BIT_LENGTH('가나다'), CHAR_LENGTH('가나다'), LENGTH('가나다');
```

- CONCAT(문자열1, 문자열2, ...), CONCAT_WS(문자열1, 문자열2, ...)
  - CONCAT( ) 함수는 문자열을 이어줌
  - CONCAT_WS( ) 함수는 구분자와 함께 문자열을 이어줌
  - 다음 쿼리문을 실행하면 구분자 /를 추가하여 '2020/01/01'이 출력

```
SELECT CONCAT_WS('/', '2020', '01', '01');
```

4. 내장 함수(문자열 함수)

- ELT(위치, 문자열1, 문자열2, ...), FIELD(찾을 _ 문자열, 문자열1, 문자열2, ...),
- FIND_IN_SET(찾을 _ 문자 열, 문자열 _ 리스트), INSTR(기준 _ 문자열, 부분 _ 문자열),
- LOCATE(부분 _ 문자열, 기준 _ 문자열)
  - ELT( ) 함수는 첫 번째 인수인 '위치'에 적힌 숫자를 보고 그 숫자 번째에 있는 문자열을 반환
  - FIELD( ) 함수는 찾을 문자열의 위치를 찾아 반환하는데, 매치되는 문자열이 없으면 0을 반환
  - FIND_IN_SET( ) 함수는 찾을 문자열을 문자열 리스트에서 찾아 위치를 반환
  - INSTR( ) 함수는 기준 문자열에서 부분 문자열을 찾아그 시작 위치를 반환
  - LOCATE( ) 함수는 INSTR( ) 함수와 동일하지만 파라미터의 순서가 반대
  - 다음 쿼리문을 실행하면 '둘, 2, 2, 3, 3'이 출력

```
SELECT ELT(2, '하나', '둘', '셋'), FIELD('둘', '하나', '둘', '셋'), FIND_IN_SET('둘', '하나,둘,셋'), INSTR('하나둘셋', '둘'), LOCATE('둘', '하나둘셋');
```

- FORMAT(숫자, 소수점_자릿수)
  - 숫자를 소수점 이하 자릿수까지 표현하고 1000단위마다 쉼표(,)를 넣음
  - 다음 쿼리문을 실행하면 '123.456.1235'가 출력

```
SELECT FORMAT(123456.123456, 4);
```

4. 내장 함수(문자열 함수)

- LOWER(문자열), UPPER(문자열)
  - 대문자를 소문자로, 소문자를 대문자로 바꿈
  - 다음 쿼리문을 실행하면 'abcdefgh'와 'ABCDEF GH'가 출력

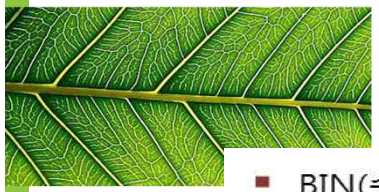
```
SELECT LOWER('abcdEFGH'), UPPER('abcdEFGH');
```

- INSERT(기준 _ 문자열, 위치, 길이, 삽입할 _ 문자열)
  - 기준 문자열의 위치부터 길이만큼을 지우고 삽입할 문자열을 끼워넣음
  - 다음 쿼리문을 실행하면 'ab@@@@ghi'와 'ab@@@@efghi'가 출력

```
SELECT INSERT('abcdefghi', 3, 4, '@@@@'), INSERT('abcdefghi', 3, 2, '@@@@');
```

- LEFT(문자열, 길이), RIGHT(문자열, 길이)
  - 왼쪽 또는 오른쪽에서 문자열의 길이만큼 반환
  - 다음 쿼리문을 실행하면 'abc'와 'ghi'가 출력

```
SELECT LEFT('abcdefghi', 3), RIGHT('abcdefghi', 3);
```



4. 내장 함수(문자열 함수)

- BIN(숫자), HEX(숫자), OCT(숫자)
  - 2진수, 16진수, 8진수의 값을 반환
  - 다음 쿼리문을 실행하면 2진수 '11111', 16진수 '1F', 8진수 '37'이 출력

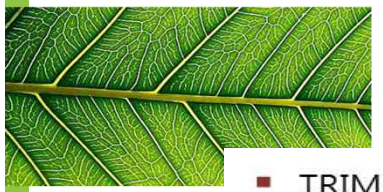
```
SELECT BIN(31), HEX(31), OCT(31);
```

- LPAD(문자열, 길이, 채울_문자열), RPAD(문자열, 길이, 채울_문자열)
  - 문자열을 길이만큼 늘리고 빈 곳을 채울 문자열로 채움
  - 다음 쿼리문을 실행하면 '###쿡북'과 '쿡 북###'가 출력

```
SELECT LPAD('쿡북', 5, '##'), RPAD('쿡북', 5, '##');
```

- LTRIM(문자열), RTRIM(문자열)
  - 문자열의 왼쪽 또는 오른쪽 공백을 제거(중간의 공백은 제거되지 않음)
  - 다음 쿼리문을 실행 하면 둘 다 공백이 제거된 '쿡북'이 출력

```
SELECT LTRIM(' 쿡북'), RTRIM('쿡북 ');
```



4. 내장 함수(문자열 함수)

- TRIM(문자열), TRIM(방향 자를_문자열 FROM 문자열)
  - TRIM( ) 함수는 문자열의 앞뒤 공백을 모두 없앰
  - '방향' 인자에는 앞을 의미하는 LEADING, 양쪽을 의미하는 BOTH, 뒤를 의미하는 TRAILING이 올 수 있음
  - 다음 쿼리문을 실행하면 '쿡북'과 '재미있어요.'가 출력

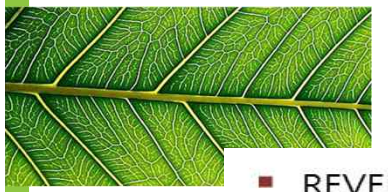
```
SELECT TRIM(' 쿡북 '), TRIM(BOTH 'ㅋ' FROM 'ㅋㅋㅋ재미있어요.ㅋㅋㅋ');
```

- REPEAT(문자열, 횟수)
  - 문자열을 횟수만큼 반복
  - 다음 쿼리문을 실행하면 '쿡북쿡북쿡북'이 출력

```
SELECT REPEAT('쿡북', 3);
```

- REPLACE(문자열, 원래 _ 문자열, 바꿀 _ 문자열)
  - 문자열에서 원래 문자열을 찾아 바꿀 문자열로 치환
  - 다음 쿼리문을 실행하면 ' IT CookBook MySQL'이 출력

```
SELECT REPLACE ('IT 쿡북 MySQL', '쿡북' , 'CookBook');
```



4. 내장 함수(문자열 함수)

- REVERSE(문자열)
  - 문자열의 순서를 거꾸로 반환
  - 다음 쿼리문을 실행하면 ' LQSyM'이 출력

```
SELECT REVERSE ('MySQL');
```

- SPACE(길이)
  - 길이만큼의 공백을 반환
  - 다음 쿼리문을 실행하면 ' IT CookBook MySQL'이 출력

```
SELECT CONCAT('IT', SPACE(10), 'CookBook MySQL');
```

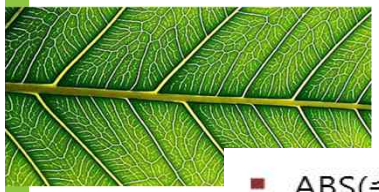
- SUBSTRING(문자열, 시작위치, 길이) 또는 SUBSTRING(문자열 FROM 시작위치 FOR 길이)
  - 시작 위치부터 길이만큼 문자를 반환
  - 길이를 생략하면 문자열의 끝까지 반환
  - 다음 쿼리 문을 실행하면 '민국'이 출력

```
SELECT SUBSTRING('대한민국만세', 3, 2);
```

4. 내장 함수(문자열 함수)

- SUBSTRING_INDEX(문자열, 구분자, 횟수)
  - 문자열에서 구분자가 왼쪽부터 '횟수' 인자에 적힌 숫자 번째 나오면 그 이후의 문자열은 버리고 앞에 있는 문자열만 출력
  - 횟수가 음수이면 오른쪽부터 세어 왼쪽의 남은 문자열을 버리고 출력
  - 다음 쿼리문을 실행하면 'www.mysql'과 'mysql.com'이 출력

```
SELECT SUBSTRING_INDEX('www.mysql.com', '.', 2), SUBSTRING_INDEX('www.mysql.com', '.', -2);
```



5. 내장 함수(수학 함수)

- ABS(숫자)
  - 숫자의 절댓값을 계산
  - 다음 쿼리문을 실행하면 절댓값인 '100'이 출력

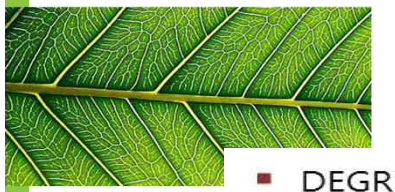
```
SELECT ABS(-100);
```

- CEILING(숫자), FLOOR(숫자), ROUND(숫자)
  - 올림, 내림, 반올림을 계산
  - 다음 쿼리문을 실행하면 '5, 4, 5'가 출력

```
SELECT CEILING(4.7), FLOOR(4.7), ROUND(4.7);
```

- CONV(숫자, 원래 _ 진수, 변환할 _ 진수)
  - 숫자를 원래 진수에서 변환할 진수로 변환
  - 다음 쿼리문을 실행하면 16진수 AA를 2진수로 변환한 '10101010'과 10진수 100을 8진수로 변환한 '144'가 출력

```
SELECT CONV('AA', 16, 2), CONV(100, 10, 8);
```



5. 내장 함수(수학 함수)

- DEGREES(숫자), RADIANS(숫자), PI( )
  - DEGREES( ) 함수는 라디안 값을 각도 값으로 변환
  - RADIANS( ) 함수는 각도 값을 라디안 값으로 변환
  - PI( ) 함수는 파이 값인 3.141592를 반환
  - 다음 쿼리문을 실행하면 파이의 각도 값인 '180'과 180의 라디안 값인 3.141592...가 출력

```
SELECT DEGREES(PI()), RADIANS(180);
```

- MOD(숫자1, 숫자2) 또는 숫자1 % 숫자2 또는 숫자1 MOD 숫자2
  - 숫자1을 숫자2로 나눈 나머지 값을 반환
  - 다음 쿼리문을 실행하면 모두 157을 10으로 나눈 나머지 값 ' 7'이 출력

```
SELECT MOD(157, 10), 157 % 10, 157 MOD 10;
```

- POW(숫자1, 숫자2), SQRT(숫자)
  - POW( ) 함수는 숫자1을 숫자2만큼 거듭제곱한 값을 반환
  - SQRT( ) 함수는 숫자의 제곱근을 반환
  - 다음 쿼리문을 실행하면 2의 3제곱인 8과 루트 9의 값인 3이 출력

```
SELECT POW(2,3), SQRT(9);
```

5. 내장 함수(수학 함수)

- RAND( )
  - 0 이상 1 미만의 실수를 구함
  - 다음 쿼리문을 실행하면 0~1 미만의 실수와 주사위 숫자가 출력

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

- SIGN(숫자)
  - 숫자가 양수, 0, 음수인지 구하여 1, 0, -1 중 하나를 반환
  - 다음 쿼리문을 실행하면 '1, 0, -1' 이 출력

```
SELECT SIGN(100), SIGN(0), SIGN(-100.123);
```

- TRUNCATE(숫자, 정수)
  - 숫자를 소수점을 기준으로 정수 위치까지 구하고 나머지는 버림
  - 다음 쿼리문을 실행하면 '12345.12'와 '12300'이 출력

```
SELECT TRUNCATE(12345.12345, 2), TRUNCATE(12345.12345, -2);
```

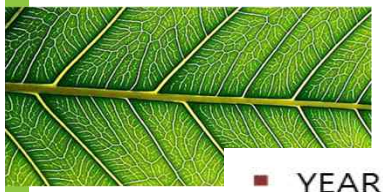
3. 내장 함수(날짜/시간 함수)

- ADDDATE(날짜, 차이), SUBDATE(날짜, 차이)
  - 날짜를 기준으로 차이를 더하거나 뺀 날짜를 반환
  - 다음 쿼리문을 실행하면 31일 후 또는 한 달후인 '2020-02-01'과 31일 전 또는 한 달 전인 '2019-12-01' 이 출력

```
SELECT ADDDATE('2020-01-01', INTERVAL 31 DAY), ADDDATE('2020-01-01', INTERVAL 1 MONTH);
SELECT SUBDATE('2020-01-01', INTERVAL 31 DAY), SUBDATE('2020-01-01', INTERVAL 1 MONTH);
```

- ADDTIME(날짜/시간, 시간), SUBTIME(날짜/시간, 시간)
  - 날짜/시간을 기준으로 시간을 더하거나 뺀 결과를 반환
  - 다음 쿼리문의 첫 번째는 1시간 1분 1초 후인 '2020-01-02 01:01:00'과 2시간 10분 10초 후인 '17:10:10' 이 출력되고, 두 번째는 1시간 1분 1초 전인 '2020-01-01 22:58:58'과 2시간 10분 10초 전인 '12:49:50' 이 출력

```
SELECT ADDTIME('2020-01-01 23:59:59', '1:1:1'), ADDTIME('15:00:00', '2:10:10');
SELECT SUBTIME('2020-01-01 23:59:59', '1:1:1'), SUBTIME('15:00:00', '2:10:10');
```



3. 내장 함수(날짜/시간 함수)

- YEAR(날짜), MONTH(날짜), DAY(날짜), HOUR(시간), MINUTE(시간), SECOND(시간), MICROSECOND(시간)
  - 날짜 또는 시간에서 연, 월, 일, 시, 분, 초, 밀리초를 구함
  - 다음 쿼리문을 실행하면 현재 연, 월, 일, 시, 분, 초, 밀리초가 출력

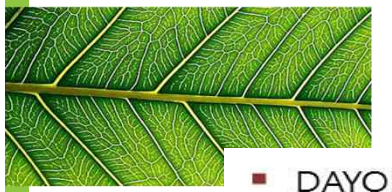
```
SELECT YEAR(CURDATE()), MONTH(CURRENT_DATE()), DAYOFMONTH(CURRENT_DATE);
SELECT HOUR(CURTIME()), MINUTE(CURRENT_TIME()), SECOND(CURRENT_TIME),
MICROSECOND(CURRENT_TIME);
```

- DATE( ), TIME( )
  - DATETIME 형식에서 연-월-일과 시:분:초만 추출
  - 다음 쿼리문을 실행하면 현재 연-월-일과 시:분:초가 출력

```
SELECT DATE(NOW()), TIME(NOW());
```

- DATEDIFF(날짜1, 날짜2), TIMEDIFF(날짜1또는 시간1, 날짜1또는 시간2)
  - DATEDIFF( ) 함수는 날짜1-날짜2의 결과를 반환
  - 다음 쿼리문을 실행하면 2023년 1월 1일에서 오늘 날짜를 뺀 일자와 '11:12:49'가 출력

```
SELECT DATEDIFF('2023-01-01', NOW()), TIMEDIFF('23:23:59', '12:11:10');
```



3. 내장 함수(날짜/시간 함수)

- DAYOFWEEK(날짜), MONTHNAME( ), DAYOFYEAR(날짜)
  - DAYOFWEEK( ) 함수는 요일(1: 일~7: 토) 반환
  - MONTHNAME( ) 함수는 월의 영문(January ~December) 반환
  - DAYOFYEAR( ) 함수는 1년 중 몇 번째 날(1~366)인지를 반환
  - 다음 쿼리문을 실행하면 현재 요일이 숫자로, 현재 월이 영문으로, 1년 중 며칠이 지났는지 숫자로 출력

```
SELECT DAYOFWEEK(CURDATE()), MONTHNAME(CURDATE()), DAYOFYEAR(CURDATE());
```

- LAST_DAY(날짜)
  - 입력한 월의 마지막 날짜를 반환
  - 다음 쿼리 문을 실행하면 '2020-02-29'가 출력

```
SELECT LAST_DAY('2020-02-01');
```

- MAKEDATE(연도, 정수)
  - 연도의 첫날부터 정수만큼 지난 날짜를 반환
  - 음 쿼리문을 실행하면 2020년 1월 1일부터 32 일이 지난 날짜인 '2020-02-01'이 출력

```
SELECT MAKEDATE(2020, 32);
```

3. 내장 함수(날짜/시간 함수)

- MAKETIME(시, 분, 초)
  - 시, 분, 초를 이용하여 '시:분:초'의 TIME 형식을 만듦
  - 다음 쿼리문을 실행하면 '12:11:10'의 TIME 형식을 출력

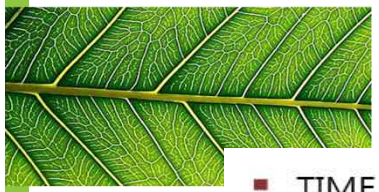
```
SELECT MAKETIME(12, 11, 10);
```

- PERIOD_ADD(연월, 개월수), PERIOD_DIFF(연월1, 연월2)
  - PERIOD_ADD( ) 함수는 연월부터 개월 수만큼 지난 연월을 반환
  - PERIOD_DIFF( )는 연월1-연월2의 개월 수를 반환
  - 다음 쿼리 문을 실행하면 '2020년 12월'과 '13개월'이 출력

```
SELECT PERIOD_ADD(202001, 11), PERIOD_DIFF(202001, 201812);
```

- QUARTER(날짜)
  - 날짜가 4분기 중에서 몇 분기인지를 반환
  - 다음 쿼리문을 실행하면 7월 7일에 해당하는 '3분기'가 출력

```
SELECT QUARTER('2020-07-07');
```

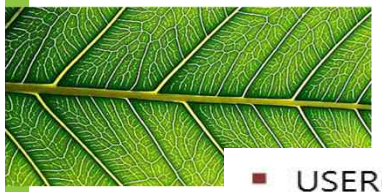


3. 내장 함수(날짜/시간 함수)

- TIME_TO_SEC(시간)
  - 시간을 초 단위로 반환
  - 다음 쿼리문을 실행하면 '43870초'가 출력

```
SELECT TIME_TO_SEC('12:11:10');
```

- CURDATE( ), CURTIME( ), NOW( ), SYSDATE( )
  - CURDATE( )는 현재 '연-월-일' 반환
  - CURTIME( )은 현재 '시:분:초' 반환
  - NOW( )와 SYSDATE( )는 현재 '연-월-일 시:분:초' 반환



3. 내장 함수(시스템/정보 함수)

- USER( ), DATABASE( )
  - 현재 사용자와 현재 선택된 데이터베이스를 반환
  - 다음 쿼리문을 실행하면 현재 사용자와 현재 선택된 데이터베이스가 출력

```
SELECT CURRENT_USER(), DATABASE();
```

- FOUND_ROWS( )
  - 바로 앞의 SELECT 문에서 조회된 행의 개수를 반환
  - 다음 쿼리문에서는 고객 테이블의 10개 행을 조회했으므로 '10'이 출력

```
SELECT * FROM userTBL;
SELECT FOUND_ROWS();
```

- ROW_COUNT( )
  - 바로 앞의 INSERT, UPDATE, DELETE 문에서 삽입, 수정, 삭제된 행의 개수를 반환
  - 다음 쿼리문은 UPDATE 문에서 구매 테이블의 12개 행을 변경했으므로 '12'가 출력

```
UPDATE buyTBL SET price=price * 2;
SELECT ROW_COUNT();
```

3. 내장 함수(시스템/정보 함수)

- SLEEP(초)
  - 쿼리의 실행을 잠깐 멈춤
  - 다음 쿼리문을 실행하면 5초 동안 멈춘 후 결과가 출력

```
SELECT SLEEP(5);
SELECT '5초 후에 이게 보여요';
```

- VERSION( )
  - 현재 MySQL의 버전을 출력


감사합니다.