

## 6. SQL - 데이터 가공

#0.강의/2.데이터베이스로드맵/1.입문

- /산술 연산
- /문자열 함수
- /NULL 함수
- /다양한 함수 소개
- /문제와 풀이
- /정리

### 산술 연산

쇼핑몰을 운영하다 보면 단순히 '어떤 상품이 얼마인가?'를 넘어 '가격과 재고 수량을 곱한 값은?', '고객 정보를 보기 좋게 합쳐서 보여줄 순 없나?'와 같은 실질적인 요구사항이 끊임없이 생긴다.

이번에는 데이터베이스에 저장된 데이터를 단순 조회하는 것을 넘어, 우리가 원하는 정보로 '가공'하고 '계산'하는 방법을 배워보자.

### 산술 연산이 필요한 이유

우리가 운영하는 쇼핑몰의 `products` 테이블에는 상품의 가격(`price`)과 재고 수량(`stock_quantity`)이 각각 저장되어 있다. 그런데 만약 **각 상품의 재고 자산 총액**이 얼마인지 알고 싶다면 어떻게 해야 할까? 재고 자산 총액은 `가격 * 재고 수량`으로 계산할 수 있다.

지금까지 배운 `SELECT` 문으로는 각 상품의 가격과 재고를 따로 조회할 수밖에 없다.

```
SELECT name, price, stock_quantity FROM products;
```

#### [실행 결과]

name	price	stock_quantity
갤럭시	10000	55
LG 그램	20000	35

아이폰	5000	55
에어팟	3000	110
보급형 스마트폰	5000	100

이 결과를 보고 각 행마다 계산기로 가격 \* 재고 수량을 직접 계산해야 할까? 데이터가 수천, 수만 개라면? 바로 이럴 때, SQL 안에서 직접 계산하는 기능이 필요하다.

## SELECT 절 안에서 사칙연산 활용하기

해결책은 아주 간단하다. SELECT 문으로 컬럼을 조회할 때, 숫자 타입의 컬럼이라면 우리가 아는 사칙연산(+, -, \*, /)을 그대로 사용할 수 있다.

각 상품의 재고 자산 총액을 계산해보자. price 컬럼과 stock\_quantity 컬럼을 곱하면 된다.

```
SELECT name, price, stock_quantity, price * stock_quantity FROM products;
```

### [실행 결과]

name	price	stock_quantity	price * stock_quantity
갤럭시	10000	55	550000
LG 그램	20000	35	700000
아이폰	5000	55	275000
에어팟	3000	110	330000
보급형 스마트폰	5000	100	500000

어떤가? 데이터베이스가 각 상품별로 price와 stock\_quantity를 곱한 결과를 새로운 컬럼으로 만들어서 보여준다. 이제 더 이상 우리가 직접 계산할 필요가 없다.

그런데 price \* stock\_quantity라는 컬럼 이름이 너무 길고 보기 좋지 않다. 이럴 때는 앞서 배운 것처럼 별칭(Alias)을 사용해서 깔끔한 이름을 붙여줄 수 있다. AS 키워드를 사용한다.

```
SELECT
  name,
  price,
  stock_quantity,
  price * stock_quantity AS total_stock_value
FROM
  products;
```

#### [실행 결과]

name	price	stock_quantity	total_stock_value
갤럭시	10000	55	550000
LG 그램	20000	35	700000
아이폰	5000	55	275000
에어팟	3000	110	330000
보급형 스마트폰	5000	100	500000

이제 `total_stock_value` 라는 훨씬 의미 있고 깔끔한 이름으로 결과를 볼 수 있다. 실무에서는 이렇게 계산된 컬럼에 별칭을 붙이는 것이 좋다.

#### 다양한 산술 연산 소개

곱셈(\*) 외에도 다른 연산들을 사용할 수 있다.

**덧셈 (+):** 모든 상품 가격에 배송비 3000원을 더해서 '예상 구매 가격'을 계산해볼 수 있다.

```
SELECT name, price, price + 3000 AS expected_price FROM products;
```

name	price	expected_price
갤럭시	10000	13000

LG 그램	20000	23000
아이폰	5000	8000
에어팟	3000	6000
보급형 스마트폰	5000	8000

**뺄셈 (-):** 모든 상품을 1000원 할인 판매한다고 가정하고 할인된 가격을 계산할 수 있다.

```
SELECT name, price, price - 1000 AS discounted_price FROM products;
```

name	price	discounted_price
갤럭시	10000	9000
LG 그램	20000	19000
아이폰	5000	4000
에어팟	3000	2000
보급형 스마트폰	5000	4000

**나눗셈 (/):** 상품 가격을 10으로 나눠서 10개월 무이자 할부 시 월 납부액을 계산할 수 있다.

```
SELECT name, price, price / 10 AS monthly_payment FROM products;
```

name	price	monthly_payment
갤럭시	10000	1000.0000
LG 그램	20000	2000.0000
아이폰	5000	500.0000
에어팟	3000	300.0000
보급형 스마트폰	5000	500.0000

이처럼 `SELECT` 절에서 산술 연산을 활용하면, 단순히 저장된 데이터를 꺼내 보는 것을 넘어, 의미 있는 정보를 계산하고 가공하여 조회할 수 있다.

## 문자열 함수

### 문자열 함수가 필요한 이유

이번에는 고객 관리 페이지를 만든다고 생각해보자. `customers` 테이블에는 고객의 이름(`name`)과 이메일(`email`)이 별도의 컬럼에 저장되어 있다.

```
SELECT name, email FROM customers;
```

#### [실행 결과]

name	email
이순신	yisunsin@example.com
세종대왕	sejong@example.com
장영실	youngsil@example.com

그런데 보고서 화면에는 다음과 같이 이름과 이메일을 합쳐서 하나의 문자열로 보기 좋게 표시하고 싶다.

```
이순신 (yisunsin@example.com)
```

이런 요구사항은 아주 흔하다. 주소도 마찬가지다. '시', '구', '상세주소'가 나뉘어 저장된 경우, "서울특별시 종로구 사직로 100" 처럼 합쳐서 보여줘야 할 때가 많다.

### 해결책: `CONCAT()` 함수로 문자열 합치기

이럴 때 사용하는 아주 유용한 함수가 바로 `CONCAT()` 함수다. `CONCAT`은 'concatenate', 즉 '연결하다'라는 뜻의 영

어 단어에서 유래했다. 이름 그대로 괄호 안에 전달된 여러 문자열을 순서대로 이어 붙여 하나의 문자열로 만들어준다.

#### ☰ 함수란?

함수는 어떤 값을 받아서 특별한 처리를 하고 그 결과를 반환하는 기능으로 이해하면 된다.

`CONCAT(string1, string2, ...)`

- `CONCAT`에는 붙이고 싶은 값을 , (쉼표)로 구분해서 넣으면 된다.

`CONCAT()` 함수를 사용해서 고객의 이름과 이메일을 합쳐보자.

```
SELECT CONCAT(name, ' (', email, ')') FROM customers;
```

#### [실행 결과]

**CONCAT(name, ' (', email, ')')**

이순신 (yisunsin@example.com)

세종대왕 (sejong@example.com)

장영실 (youngsil@example.com)

결과를 보면, `name` 컬럼 값, ( 문자, `email` 컬럼 값, ) 문자가 순서대로 합쳐져서 우리가 원하던 형태로 출력되었다. 역시 컬럼 이름이 너무 지저분하니 `AS`로 별칭을 붙여주자.

```
SELECT
  CONCAT(name, ' (', email, ')') AS name_and_email
FROM
  customers;
```

#### [실행 결과]

**name\_and\_email**

이순신 (yisunsin@example.com)

세종대왕 (sejong@example.com)

장영실 (youngsil@example.com)

- 이제 `name_and_email`이라는 깔끔한 이름으로 합쳐진 문자열을 얻을 수 있다.


## 문자열을 다루는 기본 함수 소개

`CONCAT()` 외에도 실무에서 자주 사용하는 몇 가지 유용한 문자열 함수들이 있다.

`CONCAT_WS(separator, string1, string2, ...)`

- `CONCAT`과 비슷하지만, 첫 번째 인자로 '구분자'를 받아 각 문자열 사이에 자동으로 넣어준다. `WS`는 'With Separator'의 약자다.
- 예시: 고객 이름과 이메일 주소, 주소를 `-`로 연결하기

```
SELECT CONCAT_WS(' - ', name, email, address) AS customer_details FROM customers;
```

 `CONCAT_WS()`는 MySQL 전용 함수이다. 다른 데이터베이스와 호환되지 않는다.

### [실행 결과]

#### customer\_details

이순신 - `yisunsin@example.com` - 서울특별시 중구 세종대로

세종대왕 - `sejong@example.com` - 서울특별시 종로구 사직로

장영실 - `youngsil@example.com` - 부산광역시 동래구 복천동

`UPPER()` / `LOWER()`

- 문자열을 모두 대문자 또는 소문자로 변경한다. 이메일 검색 등에서 대소문자 구분 없이 비교해야 할 때 유용하다.
- 예시: 이메일을 모두 대문자로 출력하기

```
SELECT email, UPPER(email) AS upper_email FROM customers;
```

### [실행 결과]

email	upper_email
yisunsin@example.com	YISUNSIN@EXAMPLE.COM
sejong@example.com	SEJONG@EXAMPLE.COM
youngsil@example.com	YOUNGSIL@EXAMPLE.COM

### LENGTH(), CHAR\_LENGTH()

- LENGTH() : 문자열의 길이를 바이트 단위로 반환한다.
- CHAR\_LENGTH() : 글자 수를 반환한다.
- 예시: 고객 이름의 길이 확인하기

```
SELECT name, CHAR_LENGTH(name) as char_length, LENGTH(name) AS byte_length
FROM customers;
```

### [실행 결과] (UTF-8 인코딩 기준 한글은 3바이트)

name	char_length	byte_length
이순신	3	9
세종대왕	4	12
장영실	3	9

## NULL 함수

### NULL 함수가 필요한 이유

products 테이블을 다시 한번 보자. 상품 데이터 중에 '보급형 스마트폰'은 상품 설명(description)이 NULL 값으로 들어가 있다.



```
SELECT name, description FROM products;
```

#### [실행 결과]

name	description
갤럭시	최신 AI 기능이 탑재된 고성능 스마트폰
LG 그램	초경량 디자인과 강력한 성능을 자랑하는 노트북
아이폰	직관적인 사용자 경험을 제공하는 스마트폰
에어팟	편리한 사용성의 무선 이어폰
보급형 스마트폰	NULL

만약 이 결과를 그대로 웹사이트에 표시한다면, 상품 설명 부분에 아무것도 나오지 않거나 'NULL'이라는 단어가 그대로 노출될 수 있다. 이는 사용자에게 좋지 않은 경험을 준다. 차라리 '상품 설명이 없습니다.'와 같은 안내 문구를 보여주는 것이 더 나은 선택이다.

이처럼 데이터에 존재할 수 있는 NULL 값을 그대로 노출하는 대신, 우리가 원하는 특정 값으로 대체해서 보여줘야 하는 상황이 바로 우리가 해결해야 할 문제다.

### 해결책: NULL 값을 다루는 함수들

MySQL은 NULL 값을 효과적으로 처리할 수 있는 함수들을 제공한다. 가장 대표적인 것이 IFNULL() 과 COALESCE() 이다.

IFNULL(표현식1, 표현식2)

IFNULL() 함수는 두 개의 인자를 받는다.

1. 표현식1 : NULL 인지 아닌지 검사할 컬럼이나 값.
2. 표현식2 : 표현식1 이 NULL 일 경우 대신 반환할 값.

만약 표현식1 이 NULL 이 아니면 표현식1 의 값을 그대로 반환하고, NULL 이면 표현식2 의 값을 반환한다.

`description` 컬럼의 `NULL` 값을 '상품 설명 없음'으로 바꿔보자.

```
SELECT
  name,
  IFNULL(description, '상품 설명 없음') AS description
FROM
  products;
```

### [실행 결과]

name	description
갤럭시	최신 AI 기능이 탑재된 고성능 스마트폰
LG 그램	초경량 디자인과 강력한 성능을 자랑하는 노트북
아이폰	직관적인 사용자 경험을 제공하는 스마트폰
에어팟	편리한 사용성의 무선 이어폰
보급형 스마트폰	상품 설명 없음

- 결과를 보면 '보급형 스마트폰'의 `description`이 더 이상 `NULL`이 아니라 우리가 지정한 '상품 설명 없음'으로 대체된 것을 확인할 수 있다.

`COALESCE`(표현식1, 표현식2, ...)

`COALESCE()` 함수는 `IFNULL()` 보다 조금 더 유연하다. 괄호 안에 여러 개의 인자를 전달할 수 있으며, 왼쪽부터 차례대로 확인해서 **처음으로 `NULL`이 아닌 값**을 반환한다. 모든 인자가 `NULL`이면 결국 `NULL`을 반환한다.

우리 예제처럼 단순히 `NULL`일 때 다른 값 하나로 바꾸는 상황에서는 `IFNULL()`과 `COALESCE()`의 기능이 거의 동일하다.

```
SELECT
  name,
  COALESCE(description, '상품 설명 없음') AS description
FROM
  products;
```

위 쿼리의 실행 결과는 `IFNULL()` 을 사용했을 때와 완전히 같다.

그렇다면 `COALESCE()` 는 언제 더 유용할까? 예를 들어, 상품에 `short_description` (짧은 설명) 컬럼과 `long_description` (긴 설명) 컬럼이 둘 다 있다고 가정해보자. 우리는 짧은 설명이 있으면 그것을 쓰고, 없으면 긴 설명을 쓰고, 둘 다 없으면 '설명 없음'을 표시하고 싶을 수 있다. 이럴 때 `COALESCE()` 가 아주 유용하다.

```
COALESCE(short_description, long_description, '설명 없음')
```

이런 식으로 여러 대안을 순서대로 제시할 수 있다는 점에서 `COALESCE()` 가 `IFNULL()` 보다 더 확장성이 좋다.

## 다양한 함수 소개

지금까지 우리는 특정 문제를 해결하기 위해 산술 연산자와 몇 가지 함수(`CONCAT`, `IFNULL` 등)를 배웠다. 하지만 데이터베이스, 특히 MySQL은 훨씬 더 많고 강력한 내장 함수들을 제공한다. 이 모든 것을 하나하나 외울 필요는 없다. "아, 이런 종류의 기능도 있구나" 정도만 간단히 알아두고, 필요할 때 공식 문서를 찾아보면 충분하다. 여기서는 어떤 종류의 함수들이 있는지 간단히 소개하겠다.

💡 이런 기능들은 절대로! 외우지 말자. 이런 것이 있다는 것 정도만 알아두고 필요할 때 검색하는 습관을 들이자

## SQL 표준 함수 소개

SQL 표준 함수는 대부분의 관계형 데이터베이스(MySQL, PostgreSQL, Oracle 등)에서 거의 동일하게 동작하는 함수들을 말한다. 따라서 이 함수들에 익숙해지면 다른 데이터베이스 시스템으로 전환하더라도 큰 어려움 없이 적응할 수 있다.

또한 SQL 표준 함수는 대부분의 RDBMS(관계형 데이터베이스 관리 시스템)에서 공통으로 지원하므로 데이터베이스 이전 시에도 코드 호환성을 유지하는 데 도움이 된다.

### 문자열 함수 (String Functions)

문자 데이터를 조작하고 처리한다.

함수	설명
<code>UPPER(string)</code>	문자열을 모두 대문자로 변환한다.

LOWER(string)	문자열을 모두 소문자로 변환한다.
SUBSTRING(...)	문자열의 일부를 추출한다.
TRIM(...)	문자열의 앞, 뒤 또는 양쪽에서 특정 문자나 공백을 제거한다.
POSITION(...)	문자열 내에서 특정 부분 문자열의 위치를 찾는다.
CHAR_LENGTH(string) 또는 CHARACTER_LENGTH(string)	문자열의 길이를 반환한다.

숫자 함수 (Numeric Functions)

숫자 데이터를 처리하기 위한 수학적 계산을 수행한다.

함수	설명
ABS(number)	숫자의 절댓값을 반환한다.
MOD(dividend, divisor)	나눗셈의 나머지를 반환한다.
ROUND(number, decimals)	숫자를 지정된 소수점 자릿수로 반올림한다.
CEILING(number)	숫자보다 크거나 같은 가장 작은 정수를 반환한다.
FLOOR(number)	숫자보다 작거나 같은 가장 큰 정수를 반환한다.
SQRT(number)	숫자의 제곱근을 반환한다.
POWER(base, exponent)	거듭제곱 값을 계산한다.

날짜 및 시간 함수 (Date and Time Functions)

날짜와 시간 데이터를 처리한다.

함수	설명
CURRENT_DATE	현재 날짜를 반환한다.
CURRENT_TIME	현재 시간을 반환한다.
CURRENT_TIMESTAMP	현재 날짜와 시간(타임스탬프)을 반환한다.

EXTRACT(field FROM source)	날짜/시간 값에서 특정 필드(예: YEAR, MONTH, DAY)의 값을 추출한다.
----------------------------	--

### 조건 및 변환 함수 (Conditional and Conversion Functions)

함수	설명
CAST(expression AS datatype)	표현식을 지정된 데이터 타입으로 변환한다.
COALESCE(value1, value2, ...)	인수 목록에서 첫 번째로 NULL 이 아닌 값을 반환한다.
NULLIF(expression1, expression2)	두 표현식이 같으면 NULL 을, 그렇지 않으면 첫 번째 표현식을 반환한다.

### 집계 함수 (Aggregate Functions)

여러 행의 값을 바탕으로 단일 결과 값을 계산한다. GROUP BY 절과 함께 자주 사용된다. (뒤에서 자세히 알아본다.)

함수	설명
AVG(expression)	표현식의 평균값을 반환한다.
COUNT(expression)	표현식의 결과가 NULL 이 아닌 행의 수를 반환한다.
COUNT(*)	테이블의 전체 행 수를 반환한다.
MAX(expression)	표현식의 최댓값을 반환한다.
MIN(expression)	표현식의 최솟값을 반환한다.
SUM(expression)	표현식의 합계를 반환한다.

### 데이터베이스 방언 소개

SQL은 표준이 있지만, 각 데이터베이스 제조사(Oracle, Microsoft, MySQL 등)는 자신들만의 고유한 기능과 문법을 추가했다. 이를 '데이터베이스 방언(Dialect)'이라고 부른다. 방언은 사투리다. 방언은 특정 데이터베이스에서만 동작하는 강력하고 편리한 기능을 제공하지만, 다른 데이터베이스와 호환되지 않는다는 단점이 있다. 우리가 배우는 IFNULL() 도 사실 MySQL(과 일부 DB)의 방언이고, SQL 표준에 더 가까운 것은 COALESCE() 이다.

# MySQL 전용 상세 함수

MySQL은 개발 편의성과 강력한 기능을 위해 SQL 표준 외에 다양한 함수들을 제공한다.

## 1. 문자열 함수 (String Functions)

함수	설명
CONCAT(str1, str2, ...)	둘 이상의 문자열을 합친다.
CONCAT_WS(separator, str1, ...)	지정된 구분자(separator)를 사용해 문자열들을 합친다.
GROUP_CONCAT(expression)	GROUP BY 로 그룹화된 여러 행의 문자열을 한 줄로 합친다. (집계 함수)
LPAD(str, len, padstr)	문자열(str)의 왼쪽에 padstr 을 채워 총 길이가 len 이 되도록 만든다.
RPAD(str, len, padstr)	문자열(str)의 오른쪽에 padstr 을 채워 총 길이가 len 이 되도록 만든다.
LEFT(str, len)	문자열의 왼쪽에서 len 만큼의 문자를 반환한다.
RIGHT(str, len)	문자열의 오른쪽에서 len 만큼의 문자를 반환한다.
INSTR(str, substr)	문자열(str) 내에서 부분 문자열(substr)이 처음 나타나는 위치를 반환한다. (1부터 시작)
LOCATE(substr, str, [pos])	INSTR 과 유사하지만, 인자 순서가 다르고 검색 시작 위치(pos)를 지정할 수 있다.
REPLACE(str, from_str, to_str)	문자열 내의 모든 from_str 을 to_str 로 치환한다.
FORMAT(N, D)	숫자 N 을 소수점 D 자리까지 콤마(,)로 구분하여 포맷한다.
FIND_IN_SET(str, strlist)	콤마로 구분된 문자열 리스트(strlist)에서 str 의 위치를 찾는다.
SUBSTRING_INDEX(str, delim, count)	delim 구분자를 기준으로 count 번째까지의 부분 문자열을 반환한다.

SOUNDEX(str)	문자열의 발음을 기준으로 하는 soundex 코드를 반환한다.
SPACE(N)	N 개의 공백 문자로 이루어진 문자열을 반환한다.
ELT(N, str1, str2, ...)	N 번째에 해당하는 문자열을 반환한다. (예: ELT(2, 'a', 'b', 'c') 는 'b'를 반환)
FIELD(str, str1, str2, ...)	str이 str1, str2... 목록 중 몇 번째에 있는지 위치를 반환한다.

## 2. 날짜 및 시간 함수 (Date and Time Functions)

함수	설명
NOW()	현재 날짜와 시간을 YYYY-MM-DD HH:MM:SS 형식으로 반환한다.
SYSDATE()	함수가 실행되는 시점의 정확한 날짜와 시간을 반환한다.
DATE_FORMAT(date, format)	날짜를 지정된 format 형식의 문자열로 변환한다. (예: %Y-%m-%d)
STR_TO_DATE(str, format)	format 형식의 문자열을 날짜 타입으로 변환한다.
DATEDIFF(date1, date2)	두 날짜 간의 일(day) 수 차이를 반환한다. (date1 - date2)
TIMEDIFF(time1, time2)	두 시간의 차이를 반환한다. (time1 - time2)
DATE_ADD(date, INTERVAL expr unit)	날짜에 지정된 기간(INTERVAL)을 더한다. (예: INTERVAL 1 DAY)
DATE_SUB(date, INTERVAL expr unit)	날짜에서 지정된 기간(INTERVAL)을 뺀다.
PERIOD_ADD(P, N)	YYYYMM 또는 YYMM 형식의 기간(P)에 N 개월을 더한다.
PERIOD_DIFF(P1, P2)	두 기간(P1, P2) 사이의 개월 수 차이를 반환한다.

<code>TIMESTAMPADD(unit, interval, datetime)</code>	<code>datetime</code> 에 <code>interval</code> 만큼의 <code>unit</code> (시간 단위)을 더한다.
<code>TIMESTAMPDIFF(unit, start, end)</code>	두 <code>datetime</code> 값의 차이를 <code>unit</code> (시간 단위)으로 반환한다.
<code>LAST_DAY(date)</code>	해당 날짜가 속한 달의 마지막 날을 반환한다.
<code>DAYOFWEEK(date)</code>	날짜의 요일을 숫자로 반환한다. (1:일요일, 2:월요일, ..., 7:토요일)
<code>WEEKDAY(date)</code>	날짜의 요일을 숫자로 반환한다. (0:월요일, 1:화요일, ..., 6:일요일)
<code>FROM_UNIXTIME(unix_timestamp)</code>	UNIX 타임스탬프를 날짜/시간 형식으로 변환한다.
<code>UNIX_TIMESTAMP([date])</code>	날짜를 UNIX 타임스탬프로 변환한다. (인자 없으면 현재 시간)

### 3. 숫자 함수 (Numeric Functions)

함수	설명
<code>RAND([seed])</code>	0 이상 1 미만의 무작위 실수를 반환한다. <code>seed</code> 값으로 결과를 제어할 수 있다.
<code>TRUNCATE(X, D)</code>	숫자 <code>X</code> 를 소수점 <code>D</code> 자리에서 버린다. (반올림 없음)
<code>CEIL(X)</code>	<code>CEILING</code> 과 동일. 숫자 <code>X</code> 보다 크거나 같은 가장 작은 정수를 반환한다.
<code>PI()</code>	원주율 값을 반환한다.
<code>CRC32(expr)</code>	문자열 표현식의 32비트 순환 중복 검사(CRC) 값을 계산한다.
<code>SIGN(X)</code>	숫자 <code>X</code> 가 음수면 -1, 0이면 0, 양수면 1을 반환한다.
<code>CONV(N, from_base, to_base)</code>	숫자 <code>N</code> 을 <code>from_base</code> 진수에서 <code>to_base</code> 진수로 변환한다.

### 4. 흐름 제어 함수 (Flow Control Functions)



함수	설명
IF(condition, value_if_true, value_if_false)	condition이 참이면 value_if_true를, 거짓이면 value_if_false를 반환한다.
IFNULL(expr1, expr2)	expr1이 NULL이 아니면 expr1을, NULL이면 expr2를 반환한다.

## 5. 윈도우 함수 (Window Functions) - MySQL 8.0+

OVER() 절과 함께 사용되어 행의 집합(window) 내에서 연산을 수행한다.

함수	설명
ROW_NUMBER()	파티션 내에서 순서에 따라 각 행에 고유한 순번을 부여한다. (1, 2, 3, ...)
RANK()	파티션 내에서 순위에 따라 순번을 부여한다. (동점일 경우 같은 순위, 다음은 건너뛴. 1, 2, 2, 4)
DENSE_RANK()	RANK와 유사하지만 동점일 경우 다음 순위를 건너뛰지 않는다. (1, 2, 2, 3)
NTILE(N)	파티션을 N개의 그룹으로 나누고 각 행이 어느 그룹에 속하는지 번호를 부여한다.
LAG(expression, [offset], [default])	현재 행보다 offset만큼 앞에 있는 행의 값을 가져온다.
LEAD(expression, [offset], [default])	현재 행보다 offset만큼 뒤에 있는 행의 값을 가져온다.
FIRST_VALUE(expression)	파티션에서 가장 첫 번째 행의 값을 가져온다.
LAST_VALUE(expression)	파티션에서 가장 마지막 행의 값을 가져온다.

## 6. JSON 함수 - MySQL 5.7+

JSON 데이터를 생성, 파싱, 수정하는 강력한 함수들이다.

함수	설명
----	----

<code>JSON_OBJECT(key1, val1, ...)</code>	키-값 쌍으로 JSON 객체를 생성한다.
<code>JSON_ARRAY(val1, val2, ...)</code>	값들로 JSON 배열을 생성한다.
<code>JSON_EXTRACT(json_doc, path)</code>	JSON 문서에서 <code>path</code> 에 해당하는 값을 추출한다. 단축 연산자 <code>-&gt;</code> , <code>-&gt;&gt;</code> 가 있다.
<code>JSON_SET(json_doc, path, val)</code>	JSON 문서의 특정 경로에 값을 설정(삽입 또는 수정)한다.
<code>JSON_INSERT(json_doc, path, val)</code>	JSON 문서의 특정 경로에 값을 삽입한다. (기존 값이 있으면 변경 안 함)
<code>JSON_REPLACE(json_doc, path, val)</code>	JSON 문서의 특정 경로에 있는 값을 대체한다. (기존 값이 없으면 변경 안 함)
<code>JSON_REMOVE(json_doc, path)</code>	JSON 문서에서 특정 경로의 데이터를 삭제한다.
<code>JSON_CONTAINS(target, candidate)</code>	<code>candidate</code> JSON이 <code>target</code> JSON에 포함되는지 확인한다.
<code>JSON_SEARCH(json_doc, 'one' \  'all', search_str)</code>	<code>search_str</code> 과 일치하는 값의 경로를 반환한다.
<code>JSON_KEYS(json_doc)</code>	JSON 객체의 최상위 키들을 JSON 배열로 반환한다.
<code>JSON_ARRAY_APPEND(json_doc, path, val)</code>	JSON 배열의 끝에 값을 추가한다.
<code>JSON_PRETTY(json_doc)</code>	JSON 문서를 보기 좋게 줄 바꿈과 들여쓰기를 적용하여 반환한다.
<code>JSON_VALID(val)</code>	문자열이 유효한 JSON 형식인지 확인한다. (1 또는 0 반환)

## 7. 기타 함수

함수	설명
<code>UUID()</code>	범용 고유 식별자(Universally Unique Identifier)를 생성한다.

INET_ATON(ip_address)	IPv4 주소를 숫자로 변환한다.
INET_NTOA(number)	숫자를 IPv4 주소로 변환한다.
INET6_ATON(ip_address)	IPv6 주소를 바이너리 문자열로 변환한다.
INET6_NTOA(binary_string)	바이너리 문자열을 IPv6 주소로 변환한다.
GET_LOCK(str, timeout)	명명된 락(Lock)을 획득한다. 중복 실행 방지에 사용된다.
RELEASE_LOCK(str)	명명된 락을 해제한다.
IS_FREE_LOCK(str)	명명된 락이 사용 가능한지 확인한다.
FOUND_ROWS()	SQL_CALC_FOUND_ROWS 옵션과 함께 사용되어 LIMIT 절이 없었을 경우의 총 행 수를 반환한다.

실제로는 더 많은 특수 목적의 함수들이 존재하며, 최신 버전의 MySQL이 출시될 때마다 새로운 함수들이 추가될 수 있다. 가장 정확하고 완전한 정보는 **MySQL 공식 문서**를 참고하자

## MySQL 공식 문서 - 함수

<https://dev.mysql.com/doc/refman/8.0/en/functions.html>

## 문제와 풀이

### 문제1: 상품 할인 가격 계산하기

#### [문제]

products 테이블의 모든 상품에 대해 15% 할인된 가격을 계산하고, sale\_price 라는 별칭으로 출력하라. 상품의 이름, 원래 가격(price), 그리고 할인가(sale\_price)를 함께 조회해야 한다.

#### [실행 결과]

name	price	sale_price
갤럭시	10000	8500.00
LG 그램	20000	17000.00

아이폰	5000	4250.00
에어팟	3000	2550.00
보급형 스마트폰	5000	4250.00

### [정답]

```
SELECT
  name,
  price,
  price * 0.85 AS sale_price
FROM
  products;
```

## 문제2: 고객 정보 보기 좋게 합치기

### [문제]

customers 테이블을 사용하여, 각 고객의 이름과 주소를 CONCAT\_WS() 함수를 이용해 ' - '로 연결하라. 결과 컬럼의 별칭은 customer\_info로 지정한다.

### [실행 결과]

customer_info
이순신 - 서울특별시 중구 세종대로
세종대왕 - 서울특별시 종로구 사직로
장영실 - 부산광역시 동래구 복천동

### [정답]

```
SELECT
    CONCAT_WS(' - ', name, address) AS customer_info
FROM
    customers;
```

### 문제3: 상품 설명이 없는 경우 처리하기

#### [문제]

products 테이블에서 상품 설명(description)이 없는(NULL) 상품은, 상품 이름(name)을 대신 설명으로 사용하도록 조회하라. COALESCE() 함수를 사용하고, 결과 컬럼의 별칭은 product\_display\_info로 지정한다. 상품의 원래 이름과 최종적으로 표시될 정보를 함께 출력하라.

#### [실행 결과]

name	product_display_info
갤럭시	최신 AI 기능이 탑재된 고성능 스마트폰
LG 그램	초경량 디자인과 강력한 성능을 자랑하는 노트북
아이폰	직관적인 사용자 경험을 제공하는 스마트폰
에어팟	편리한 사용성의 무선 이어폰
보급형 스마트폰	보급형 스마트폰

#### [정답]

```
SELECT
    name,
    COALESCE(description, name) AS product_display_info
FROM
    products;
```

## 문제4: 여러 후보 값 중 유효한 값 선택하기

### [문제]

products 테이블에 description (상품 설명) 컬럼과 name (상품명) 컬럼이 있다. 상품 정보를 표시할 때, description 값이 존재하면 그 값을 사용하고, description이 NULL 이면 name 값을 사용하고, 만약 name 값도 NULL 이라면(그런 경우는 없지만 가정) '정보 없음'이라는 문구를 출력하고 싶다. 이 로직을 COALESCE() 함수를 사용하여 display\_text 라는 별칭으로 출력하라. 실행 결과를 참고해라.

### [실행 결과]

name	description	display_text
갤럭시	최신 AI 기능이 탑재된 고성능 스마트폰	최신 AI 기능이 탑재된 고성능 스마트폰
LG 그램	초경량 디자인과 강력한 성능을 자랑하는 노트북	초경량 디자인과 강력한 성능을 자랑하는 노트북
아이폰	직관적인 사용자 경험을 제공하는 스마트폰	직관적인 사용자 경험을 제공하는 스마트폰
에어팟	편리한 사용성의 무선 이어폰	편리한 사용성의 무선 이어폰
보급형 스마트폰	NULL	보급형 스마트폰

### [정답]

```
SELECT
  name,
  description,
  COALESCE(description, name, '정보 없음') AS display_text
FROM
  products;
```

## 문제5: 이메일 주소 분리 및 분석하기

### [문제]

customers 테이블의 email 컬럼에서 아이디 부분만 추출하고, 아이디의 글자 수를 계산하라. 아이디는 @ 앞부분의 문자열이다. SUBSTRING\_INDEX() 와 CHAR\_LENGTH() 함수를 활용하여 user\_id와 id\_length라는 별칭으로 결과를 출력하라. 다음 실행 결과를 참고하자. SUBSTRING\_INDEX(...)의 사용법을 웹 사이트에 검색해서 풀어보자.

### [실행 결과]

email	user_id	id_length
sejong@example.com	sejong	6
yisunsin@example.com	yisunsin	8
youngsil@example.com	youngsil	8

### [정답]

```
SELECT
  email,
  SUBSTRING_INDEX(email, '@', 1) AS user_id,
  CHAR_LENGTH(SUBSTRING_INDEX(email, '@', 1)) AS id_length
FROM
  customers;
```

## 정리

### 산술 연산

- 데이터베이스에 저장된 숫자 데이터를 조회할 때 사칙연산(+, -, \*, /)을 사용하여 원하는 값으로 가공할 수 있다.

- `SELECT` 절에서 숫자 타입의 컬럼에 연산자를 적용하여 새로운 정보를 계산한다.
- 예를 들어 `가격 * 재고 수량`으로 재고 자산 총액을 계산할 수 있다.
- 계산된 컬럼은 `AS` 키워드를 사용하여 의미 있는 별칭을 붙여주는 것이 좋다.

## 문자열 함수

- `CONCAT()` 함수는 괄호 안에 전달된 문자열이나 컬럼 값을 순서대로 이어 붙인다.
- `CONCAT_WS(구분자, 문자열1, 문자열2)`는 각 문자열 사이에 지정된 구분자를 넣어 합친다.
- `UPPER()`와 `LOWER()` 함수는 문자열을 각각 대문자나 소문자로 변환한다.
- `CHAR_LENGTH()`는 글자 수를 반환하고 `LENGTH()`는 바이트 수를 반환한다.

## NULL 함수

- 데이터가 비어있음을 의미하는 `NULL` 값을 다른 값으로 대체하여 출력할 필요가 있다.
- `IFNULL(컬럼, 대체값)` 함수는 해당 컬럼이 `NULL` 일 경우 지정된 대체값을 반환한다.
- `COALESCE(값1, 값2, ...)` 함수는 나열된 값들 중 처음으로 `NULL`이 아닌 값을 반환한다.
- `COALESCE`는 여러 대안을 순서대로 검사할 수 있어 `IFNULL`보다 확장성이 좋다.

## 다양한 함수 소개

- SQL에는 데이터를 가공하는 다양한 내장 함수가 있으며 모두 외울 필요 없이 필요할 때 찾아 쓰는 것이 중요하다.
- SQL 표준 함수는 대부분의 데이터베이스에서 공통으로 지원하여 호환성이 높다.
- 데이터베이스 방언은 특정 데이터베이스 제조사에서만 제공하는 고유한 함수나 문법을 의미한다.
- MySQL은 문자열, 날짜, 숫자, 흐름 제어, JSON 등 다양한 목적의 전용 함수를 제공한다.