Шпаргалка по стандартным функциям SQL. Часть 1

ТЕКСТОВЫЕ ФУНКЦИИ

КОНКАТЕНАЦИЯ

Используй оператор || для конкатенации двух строк:

```
SELECT 'Hi ' || 'there!';
-- result: Hi there!
```

Помните, что вы можете объединять только символьные строки, используя ||. Попробуйте такой трюк с числами:

```
SELECT '' || 4 || 2;
-- result: 42
```

В некоторые базы данных встроены нестандартные решения для конкатенации строк, например `CONCAT()` или `CONCAT_WS()`. Ознакомьтесь с документацией используемой базы данных.

ОПЕРАТОР LIKE – ПАТТЕРНЫ

Используй символ _ символ для замены любого одиночного символа. Используй символ % для замены любым количеством символов (включая отсутствие символов).

Получить все имена, начинающиеся с любой буквы, за которой следует `'atherine'`:

```
SELECT name
FROM names
WHERE name LIKE '_atherine';
```

Получить все имена, которые заканчиваются на `'a'`:

```
SELECT name
FROM names
WHERE name LIKE '%a';
```

ПОЛЕЗНЫЕ ФУНКЦИИ

Получить количество символов в строке:

```
SELECT LENGTH('analysis-guide.tech');
-- result: 19
```

Преобразовать все буквы в нижний регистр:

```
SELECT LOWER('ANALYSIS-GUIDE.TECH');
-- result: analysis-guide.tech
```

Преобразовать все буквы в верхний регистр:

```
SELECT UPPER('Analysis-Guide.tech');
-- result: ANALYSIS-GUIDE.TECH
```

Преобразовать все буквы в нижний регистр и все первые буквы в верхний регистр (не реализовано в MySQL и SQL Server):

```
SELECT INITCAP('edgar frank ted cODD');
-- result: Edgar Frank Ted Codd
```

Получить только часть строки:

```
SELECT SUBSTRING('analysis-guide.tech', 9);
-- result: -guide.tech
SELECT SUBSTRING('guide.tech', 0, 6);
-- result: guide
```

Заменить часть строки:

```
SELECT REPLACE('analysis-guide.tech', '-',
'_');
-- result: analysis_guide.tech
```

ЧИСЛОВЫЕ ФУНКЦИИ

БАЗОВЫЕ ОПРЕРАЦИИ

Используй +, −, *, / чтобы выполнить базовые математические операции.

Получить количество секунд в неделе:

```
SELECT 60 * 60 * 24 * 7; -- result: 604800
```

ПРИВЕДЕНИЕ

Время от времени нужно менять тип чисел. Функция `CAST()` поможет вам в этом. Он позволяет изменить тип значения практически на любой (`integer`, `numeric`, `double precision`, `varchar`, и многие другие).

Получить целое число (без округления):

```
SELECT CAST(1234.567 AS integer);
-- result: 1234
```

Изменить тип на `double precision`:

```
SELECT CAST(column AS double precision);
```

Полезные функции

Получить остаток от деления:

```
SELECT MOD(13, 2);
-- result: 1
```

Округлить число до ближайшего целого числа:

```
SELECT ROUND(1234.56789);
-- result: 1235
```

Округлить число до трех знаков после запятой:

```
SELECT ROUND(1234.56789, 3);
-- result: 1234.568
```

Округлить число в большую сторону:

```
SELECT CEIL(13.1);  -- result: 14
SELECT CEIL(-13.9);  -- result: -13
```

Функция `CEIL(x)` возвращает наименьшее целое число не меньше x. В SQL Server эта функция называется `CEILING()`.

Округлить число в меньшую сторону:

```
SELECT FLOOR(13.8);  -- result: 13
SELECT FLOOR(-13.2);  -- result: -14
```

Функция `FLOOR(x)` Функция возвращает наибольшее целое число, не превышающее x.

Округлить ближе к 0 независимо от знака числа:

```
SELECT TRUNC(13.5);  -- result: 13
SELECT TRUNC(-13.5);  -- result: -13
```

`TRUNC(x)` аналогично функции `CAST(x AS integer)`. В MySQL, функция называется `TRUNCATE()`.

Получить абсолютное значение числа:

```
SELECT ABS(-12);  -- result: 12
```

Получить квадратный корень числа:

```
SELECT SQRT(9);  -- result: 3
```

ПРОВЕРКА ПУСТЫХ ЗНАЧЕНИЙ

Получить все строки с отсутствующим значением в поле **price:**

```
WHERE price IS NULL
```

Получить все строки с заполненным полем **weight:**

```
WHERE weight IS NOT NULL
```

Почему бы не использовать выражения `price = NULL` or `weight != NULL`? Потому что базы данных не знают, истинны ли эти выражения или ложны– они оцениваются как NULLs. Более того, если вы используете функцию или конкатенацию в столбце с **NULL** в некоторых строках, тогда результат будет возвращать также **NULL**. Вгляните:

domain	LENGTH(domain)
LearnSQL.com	12
LearnPython.com	15
NULL	*NULL*
vertabelo.com	13

ПОЛЕЗНЫЕ ФУНКЦИИ

COALESCE(x, y, ...)

Чтобы заменить NULL в запросе на что-то значимое:

```
SELECT
  domain,
COALESCE(domain, 'domain missing') FROM
contacts;
```

domain	coalesce
LearnSQL.com	LearnSQL.com
NULL	domain missing

Функция `COALESCE()` принимает любое количество аргументов и возвращает значение первого аргумента, который не NULL.

NULLIF(x, y)

Чтобы уберечь себя от ошибок деления на 0:

```
SELECT
  last_month,
  this_month,
  this_month * 100.0
    / NULLIF(last_month, 0)
    AS better_by_percent
FROM video_views;
```

last_month	this_month	better_by_percent
723786	1085679	150.0
0	178123	*NULL*

Функция `NULLIF(x, y)` возвращает NULL если x совпадает с y, иначе он вернет значение x.

CASE WHEN

Конструкция `CASE WHEN` проверяет - равны ли значения поля указанным в конструкции (например если комиссия равна 50, то возвращается значение `'normal'`). Если значение поля не совпадает со значением в `CASE WHEN`, тогда будут возвращены значения, которые указаны в `ELSE` (например если коммиссия равна 49, то будет вовзращено значение `'not available'`).

```
SELECT
  CASE fee
    WHEN 50 THEN 'normal'
    WHEN 10 THEN 'reduced'
    WHEN  0 THEN 'free'
    ELSE 'not available'
              END AS tariff
            FROM ticket_types;
```

Самый популярный тип — **поисковый CASE WHEN** – он позволяет передать условия (как бы вы написали их в условии `WHERE`), оценивает их по порядку, а затем возвращает значение для первого выполненного условия.

```
SELECT
  CASE
    WHEN score >= 90 THEN 'A'
    WHEN score >  60 THEN 'B'
    ELSE 'F'
  END AS grade
              FROM test_results;
```

Здесь все студенты, набравшие не менее 90 баллов - получат A, те, кто набрал больше 60 (и меньше 90), получат B, а остальные получат F.

РЕШЕНИЕ ПРОБЛЕМ

Целочисленное деление

Если вы не видите ожидаемых знаков после запятой, это означает что вы делите значения с типами **integer.** Приведите одно значение к **demical:** `CAST(123 AS decimal) / 2`

Деление на ноль

Чтобы избежать этой ошибки, убедитесь, что знаменатель не равен 0. Вы можете использовать функцию `NULLIF()` чтобы заменить 0 на NULL, что приведет к NULL для всего выражения: `count / NULLIF(count_all, 0)`

Неточные вычисления

Если вы делаете вычисления, используя тип real (с плавающей точкой), то в итоге получите результат с некоторой погрешностью. Все потому, что этот тип предназначен для «научных» вычислений, таких как скорость. Каждый раз, когда вам нужны результаты вычислений без погрешности (напр., если вы имеете дело с денежными расчетами), вам следует использовать тип decimal или numeric (так же подойдёт тип money, если он поддерживается).

Ошибки при округлении с указанной точностью

Большинство баз данных не будут возвращать ошибку, но если будут, проверьте документацию. Например, если вы хотите указать точность округления в PostgreSQL, значение должно быть с типом `numeric`.