Literals

In almost any program or data analysis script, you need to operate constant values called **literals**.

For example, if you're analyzing census data and need to extract census rows according to specific criteria, you often have to use literals.

String literals

A string constant in SQL is a sequence of characters surrounded by single (') or double (") quotes, for example, 'Hyperskill', 'Hello world!', and "SQL (Structured Query Language)".

To include a single-quote character within a string literal wrapped in single quotes, type two adjacent single quotes, for example, 'Jessie''s birthday'. Alternatively, wrap the literal in double quotes so that a single quote will be treated as a regular symbol. For example, "Sophie's choice".

Numeric literals

Numeric literals can be either positive or negative numbers specified as an **integer** (for example, 1, +9000, -256), **decimal** (for example, 2.3, +876.234, -1024.0), or real values in **exponential** notation (for example, 0.4e3, +7.192834e+10, -4.0e-5).

If you do not specify the sign, then a positive number is assumed by default.

Numeric literals may be INTEGER, REAL, or DECIMAL.

The data management system automatically defines a literal's type based on the context.

For example, if you specify a numeric value without a decimal point, which fits the INTEGER range of values, the system will treat it as INTEGER, and otherwise as DECIMAL.
Numeric values specified in exponential notation are treated as REAL data.
You can directly specify the type of a literal using the CAST(value AS type) function. Instead of the placeholder value and type, you can use your literal and type.

```
SELECT
   CAST(1 AS DECIMAL(20,3));
```

In the example above, the numeric value 1 is interpreted as DECIMAL (20,3) and 1.000 as a result of the query. This is because 3 as the second argument says that 3 decimal places will be outputed, even if they are zeros.

Boolean literals

Boolean literals are boolean logic truth values: TRUE and FALSE.

No matter how you specify the value (TRUE or true), these values are identical boolean literals. The same applies to the FALSE values.

Hello, World

Now, we are ready to write a traditional "Hello, World!" program. This SQL code (actually, a single query) implements it:

```
SELECT 'Hello, World!';
```

The query evaluation result is the following. Its actual representation may be different depending on the environment you run it in:

```
Hello, World!
```

Actually, the query declaratively states that we want to select this string as a result.

The statement consists of three parts: the keyword SELECT (case insensitive), the literal that we want to receive, and a semicolon that defines the end of the query.

Summarizing, a simple SQL query that extracts any literal, whether it's a string, numeric, or boolean, looks as follows. You may replace literal with any correctly specified constant you want:

```
SELECT literal;
```