Different SQL engines accept different character sets for string data. Two common sets are Unicode and ASCII, but there are other character sets as well.

Character Sets

The acronym [ASCII](https://www.w3schools.com/charsets/ref_html_ascii.asp)​ (pronounced “AS-key”) stands for “American Standard Code for Information Interchange” and dates back to the 1960s. It was the first character set widely used on computers and consists of 128 characters, including the English alphabet (both uppercase and lowercase letters), numbers, and punctuation marks. Each character is assigned a number, from 0 to 127.

The extended ASCII set adds another 127 characters, for a total of 256. It includes diacritic characters (such as é, ç, or ö) and some other common symbols, such as the degree symbol (°).

Unicode is a newer industry standard with well over 130,000 characters. It is a superset of the ASCII set; that is, Unicode characters 0–127 are the same as the ASCII characters with the same numbers. Unicode includes not only diacritic characters (such as é, ç, or ö), it also includes completely different character sets, such as Cyrillic and Brahmic language characters, and symbols such as mathematics symbols, advanced punctuation, and dingbats.

Here are some examples of Unicode characters in a literal string, using different languages:

**'Bishop à G5'**(French)

**'Бишоп до Г5'**(Macedonian)

**'ប៊ីស្សពដើម្បី G5'**(Khmer)

**'♗to G5'**(English, using a dingbat for *bishop*)

Support in SQL Engines

Most major SQL engines allow the use of Unicode characters in character strings. These can be values within a string column or field, or literal strings. *However, using Unicode characters in identifiers (such as column names or aliases) is inadvisable, even if allowed.*

With some SQL engines, you might need to modify a setting to enable handling of Unicode characters, but even with handling enabled, there can be limitations or special considerations to using these in your data. For example, some systems may require including the letter N before a literal value (for example, **N'Бишоп до Г5'**), to alert the system that the following string might use Unicode characters (or even characters not supported by the Unicode set). Other systems require that the string be enclosed in double quotes (not single quotes). For data values, some (such as Microsoft SQL Server\*) require using separate data types that allow Unicode characters.

The SQL engines available on the VM all have limited support for Unicode characters within literal strings and the standard string variable types. For example, string functions might work with non-ASCII Unicode characters, but they might not. The function **lower('ÉTIENNE')**will produce **'étienne'**with Apache Hive and PostgreSQL, but **'Étienne'**with Apache Impala and **'��tienne'**with MySQL (using the command line—see below).

Impala and MySQL treat each Unicode character as a byte array, so one character will appear to be more than one. They incorrectly identify **length('ÉTIENNE')**as 8 (treating É as two characters). Also, **substr('ÉTIENNE',1,1)**will instead return �, but **substr('ÉTIENNE',1,2)**will return **'É'**. Hive and PostgreSQL will handle both of these instances correctly.

The MySQL editor in Hue on the VM has some difficulty with Unicode characters, particularly when they appear in a column heading above a result set. You may, however, use MySQL on the command line on the VM if you are familiar with it (use **mydb**as the database, with user **training**and password **training**).

If there is a possibility that your data will include Unicode characters, check what your system’s support for Unicode is. Test your SQL statements with values that include Unicode characters so that you are aware of any issues that might arise.

\* Collation and Unicode Support. Retrieved from https://docs.microsoft.com/en-us/sql/relational-databases/collations/collation-and-unicode-support?view=sql-server-2017on August 17, 2018.