

Simple Queries In SQL

SQL, Structured Query Language, was developed in 1970s in an IBM Laboratory. SQL, some times also referred to as SEQUEL is a 4th generation non-procedural language. SQL, enables the following : (i) Creating/Modifying a database's structure (ii) Changing security settings for system (iii) Permitting users for working on databases or tables (iv) Querying database (v) Inserting/ Modifying/Deleting the [www.cbsepythonedu.com] database contents.

Some MySQL SQL Elements

The MySQL implementation of SQL has certain elements that play an important role in defining/ querying a database.

These basic [www.cbsepythonedu.com] elements are :

(i) Literals (ii) Datatypes (iii) Nulls (iv) Comments

Literals

Literals, in general, refer to a fixed data value. This fixed data value may be of character type or numeric literal. For example, 'Synthia', 'Ekagra', 'Raunak Raj Singh', 'g' and '305' are all character text literals. Numbers that are not enclosed in quotation marks are numeric literals e.g., 22, 18, 1997, 2003 are all numeric literals. Numeric literals can either be integer literals i.e., without any decimal or be real literals i.e., with a decimal point e.g., 17 is an integer literal but 17.0 are real [www.cbsepythonedu.com] literals.

Data Types

Data types are means to identify the type of data and associated operations for handling it. A values's datatype associates a fixed set of properties with the value. These properties cause different treatment of datatypes. For example, you can add values of NUMBER datatypes, but you cannot add values of CHAR or VARCHAR types. Internet datatypes supported by MySQL include the ones given here.

MySQL uses many different data types, divided into three categories.

* Numeric * Date and time, and * String types.

Numeric Data Types

MySQL uses all the standard ANSI SQL numeric data types. The following list shows the common numeric data [www.cbsepythonedu.com] types and their descriptions.

INT	A normal-sized that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 11 digits.
TINYINT	A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify [www.cbsepythonedu.com] a width of up to 4 digits.
SMALLINT	A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can

python :

specify a width of up to 5 digits.

MEDIUMINT A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the allowable range is from 0 to 16777215. You can specify width of up to 9 digits.

BIGINT A large integer that can be signed or unsigned. If signed, the allowable range is from -9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 11 [www.cbsepythonedu.com] digits.

FLOAT (M,D) A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places for a FLOAT.

DOUBLE(M,D) A Double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4 where 4 is the number of decimal. Decimal precision can go to 53 place for a DOUBLE. Real is a synonym for [www.cbsepythonedu.com] DOUBLE.

DECIMAL (M,D) An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required. NUMERIC is a synonym for [www.cbsepythonedu.com] DECIMAL.

Date and Time Types

The MySQL date and time datatypes are :

DATE A date in YYYY-MM-DD format, between 1000-01-01 and 9999-12-31 for example, December 3th, 1973 would be stored as 1973-12-30

DATETIME A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01-01 and 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored [www.cbsepythonedu.com] as 1973-12-30 15:30:00.

TIME Store the time in HH:MM:SS format.

YEAR (M) Store a year in 2 digit or 4 digit format. If the length is specified as 2 (for example YEAR(2)), YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

String/Text Types

Most data that you use in a database is in string format. This list describes the common string [www.cbsepythonedu.com] datatypes in MySQL.

python :

CHAR(M)	A fixed-length string between 1 and 255 characters in length (for example CHAR(5), right-padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
VARCHAR(M)	A variable-length string between 1 and 255 character in length; for example VARCHAR (25). You must decide a length when creating a VARCHAR (www.cbsepythonedu.com) field.
BLOB or TEXT	A field with a maximum length of 65535 character. BLOBs are “Binary Large Objects” and are used to store large amounts of binary data, such as images or other types of files. Fields defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT field. You do not specify a length with BLOB or TEXT.
TINYBLOB OR TINYTEXT	A BLOB or TEXT column with a maximum length of 235 characters. You do not specify a length with TINYBLOB or TINYTEXT.
MEDIUMBLOB or MEDIUMTEXT	A BLOB or TEXT column with a maximum length of 16777215 character. You do not specify a length with MEDIUMBLOB or (www.cbsepythonedu.com) MEDIUMTEXT
LOB or LONGTEXT	A BLOB or TEXT column with a maximum length of 4294967295 characters. You do not specify a length with LOB or LONGTEXT.
ENUM	An enumeration is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain “A” or “B” or “C” you would define your ENUM as ENUM (‘A’, ‘B’, ‘C’) and only those (www.cbsepythonedu.com) values (or NULL) could ever populate the field.

Difference between Char and Varchar Datatypes

A field is a data element which can store one type of information which is composed of characters. Number of characters that a field can store, is called field length or field width. As each character requires one byte for its storage, number of characters inside a field determine its field width in (www.cbsepythonedu.com) bytes. There can fixed length fields as well as variable length fields. Fixed length fields have fixed lengths i.e., they occupy fixed number of bytes for every data element they store. These number of bytes are determined by maximum number of characters the field can store. In fixed length fields, some space is always wasted as all data elements do not use all the space reserved but processing is much simpler in case of fixed length fields.

Variable Length fields have varied field lengths i.e., field length i.e., field length is determined separately for every data element inside the field. The number of characters in the data element become its (www.cbsepythonedu.com) field length.

The difference between CHAR and Varchar is that of fixed length and variable length. The Char

python :

datatype specifies a fixed length character string. When a column is given datatype as CHAR(n), then MySQL ensures that all values stored in that column have this length i.e., n bytes. If a value is shorter than this length n then blanks are added, but the size of value remains n bytes. VARCHAR, on the other hand, specifies a variable length string. When a column is given datatype as VARCHAR(n), then the maximum size of a value in this column can have is n byte. Each value that is stored in this column stores exactly as you specify it i.e., no blanks are added if the length is shorter than maximum length n. However, if you exceed the maximum length n, then an error message [www.cbsepythonedu.com] is displayed.

Null Values

If a column in a row has no value, then column is said to be null, or to contain a null. Nulls can appear in columns of any data type provided they are not restricted by NOT NULL or PRIMARY KEY integrity [www.cbsepythonedu.com] constraints.

Comments

A comment is a text that is not executed ; it is only for documentation purpose. A comment generally describes the statement's purpose with in an application. You can include a comment in a statement using either of these means:

* Begin the comment with /*. Proceed with the text of the comment. This text can span multiple lines. End the comment with */.

* Begin the comment with -- (followed by a space). Proceed with the text of the comment. This text cannot extend to a new line. End the comment with a line break.

* Begin the comment with #. Proceed with the text of the comment. This text cannot extend to a new line. End the comment with a line break.

A SQL statement can contain multiple comments of both styles.

SQL Command Syntax

The SQL provides a predefined set of commands that help us work on relational databases. But before discussing them, we must be familiar with the conventions and terminology used in SQL commands. Keywords are words that have a special meaning in SQL. They are understood to be instructions. Here the SQL keywords have been printed in capital letter. COmmands, or State ments, are instructions given by you to a SQL database. Commands consist of one or more logically distinct parts called clauses. Clauses begin with a keyword for which they are generally named, and consist of keywords and arguments.

Objects are structures in the databases that are given names and stored in memory. They include base tables, views, and indexes.

Symbols Used in Syntax Statements.

Symbol	Meaning
	This is a symbolic way of saying “or” that is whatever precedes this symbol may optionally be replaced by whatever follows it.
{ }	Everything enclosed in it, is treated as a unit for the purposes of

python :

	evaluating ,... or other symbols.
[]	This means, everything enclosed in it is optional
...	This means whatever precedes it may be repeated any number of times
,...	Whatever precedes this, may be repeated any number of times with the individual occurrences separated by commas.
< >	SQL and other special terms are in angle brackets.

Accessing Database

Before you start making queries upon the data in tables of a database, you need to open the database for use. For this, after logging into MySQL, you need to issue a command.

```
use<databasename> ;
```

The Select Command

The SELECT command of SQL lets you make queries on the database. A query is a command that is given to produce certain specified information from the database table (s). There are various ways and combinations, a SELECT statement can be used into. It can be used to retrieve a subset of rows or columns from one or more tables. In its simplest form, SELECT statement is used as given [www.cbsepythonedu.com] below :

```
SELECT <column name> [,<column name>,...]  
FROM <table name> ;
```

Eliminating Redunant Data (with Keyword Distinct)

By default, data is selected from all the rows of the table, even if the data appearing in the result gets duplicated. The DISTINCT keyword eliminates dulpicate rows from the results of the SELECT [www.cbsepythonedu.com] statement.

To do so, we shall write

```
SELECT DISTINCT city  
FROM Suppliers ;
```

Selecting from all the Rows All Keyword

If in place of DISTINCT, you give ALL then the result retains the duplicate output rows. It is just the same as when you specify neither DISTINCT nor ALL ;

```
SELECT ALL city FROM suppliers ;
```

Viewing Structure of Table

You have learnt just back that Use <databasename> ; opens a database for use. To view the tables in a database, you may write following command after opening the database.

```
SHOW TABLES
```

If you want to know the structure of a table, you can use Describe or Desc command as per [www.cbsepythonedu.com] following syntax :

python :

DESCRIBE | DESC <table name> ;

Using Column Aliases

The columns that you select in a query can be given a different name i.e., column alias name for output purposes. For instance, if you want to extract information (data and type) from Event table and want the name of Type column should appear as Event Type, then you should give query as [www.cbsepythonedu.com] follows :

```
SELECT date, type As "Event Type"  
FROM Event ;
```

Handling Nulls

As already mentioned, the empty values are represented as Nulls in a table. When you list a column having null values, only non-null values are displayed. For instance, if you write a query as :

```
SELECT name, birth, death  
FROM pet ;
```

If you want to substitute null with a value in the output, you can use IFNULL () function. IFNULL () function may be used as per following syntax :

Syntax:

```
IFNULL (<columnname>, value-to-be-substituted)
```

That is, if you want substitute value alive for null in death, the query should look like :

```
SELECT name, birth, IFNULL (death, "alive")  
FROM pet ;
```

You may also use column alias to replace output heading. This can be done as follows :

```
mysql> SELECT name, birth, IFNULL (death, "Alive") As "Died on"  
FROM pet ;
```

Putting Text in the Query Output

The previous example can be refined by marking the commissions as percentages with the percent sign (%). This enables you to put such items as symbols and comments in the output, as in the [www.cbsepythonedu.com] following example :

```
SELECT salesman_name, comm * 100, '%'
```

Selecting Specific Row WHERE clause

In real life, tables can contain unlimited rows. There is no need to view all the rows when only certain rows are needed. SQL enables you to define criteria to determine which rows are selected for output. The WHERE clause in SELECT statement specifies the criteria for selection of rows to be returned. The SELECT statement with WHERE clause takes the following general form :

```
SELECT <column name> [, <column name>, ...]  
FROM <table name>
```

python :

WHERE <condition>

Relational Operators

To compare two values, a relational operator is used. The result of the comparison is true or false. The SQL recognizes following relational operators :

=, >, <, >=, <=, <> (not equal to)

In CHARACTER data type comparisons, < means earlier in the alphabet and > means later in the [www.cbsepythonedu.com] alphabet.

Logical Operators

The logical operators OR (||), AND (&&) and NOT (!) are used to connect search conditions in the WHERE clause. When all the logical operators are used together, the order of precedence is NOT (!), AND (&&), and OR (||). However, parentheses can be used to override the default [www.cbsepythonedu.com] precedence.

Condition based on a Range

The BETWEEN operator defines a range of values that the column values must fall into make the condition true. The range includes both lower value and the upper value. For example, to list the items whose QOH falls between 30 to 50 (both inclusive) the command would be :

```
SELECT icode, descp, QOH
FROM items
WHERE QOH BETWEEN 30 AND 50 ;
```

The operator NOT BETWEEN is reverse of BETWEEN operator, that is the rows not satisfying the BETWEEN condition are retrieved. For example,

```
SELECT icode, descp
FROM items
WHERE ROL NOT BETWEEN 100 AND 1000 ;
```

Condition Based on a List

To specify a list of values, IN operator is used. The IN operator selects values that match any value in a given list of values.

```
SELECT * FROM members
WHERE city IN ('DELHI', 'MUMBAI', 'CHENNAI', 'BANGALORE');
```

The [www.cbsepythonedu.com] NOT IN operator find rows that do not match in the list. So if you write

```
SELECT * FROM members
WHERE city NOT IN ('DELHI', 'MUMBAI', 'CHENNAI');
```

Condition Based on Pattern Matches

SQL also includes a string-matching operator, LIKE, for comparisons on character strings using patterns. Patterns are described using two special wildcard characters :

python :

- * percent (%) . The % character matches any substrings
- * underscore (_) The _ character matches any character

To illustrate pattern matching, consider the following examples:

- * 'San%' matches any string beginning with 'San'
- * '%idge%' matches any string containing 'idge' as a substring, for example, 'Ridege', 'Bridges',
[www.cbsepythonedu.com] 'Cartidge', 'Ridegeway' etc.
- * '____' matches any string of exactly 4 characters.
- * '____%' matches any string of at least 3 character.

The LIKE keyword is used to select row containing columns that match a wildcard pattern. The keyword NOT LIKE is used to select rows that do not match the specified pattern of character. In order for patterns to include the special pattern characters (that is, %, _), SQL allows the specific of an escape character. The escape character is used immediately before a special pattern character to indicate that the special pattern character is to be treated as a normal character. We define the escape character for a LIKE comparison using the ESCAPE keyword.

To illustrate, consider the following patterns which use a backslash (\) as the escape character.

- * LIKE 'wx \%yz%' ESCAPE '\' matches all strings beginning with 'wx%yz'.
- * LIKE 'wx \\yz%' ESCAPE '\' matches all strings beginning with 'wx\yz'.

Searching For NULL

The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause. (Relational operators like =, <> etc. can't be used with NULL). For example, to list details of all [www.cbsepythonedu.com] employees whose departments contain NULL (i.e., no value), you use the command :

```
SELECT empno, empname, job
FROM emp
WHERE DeptNo IS NULL ;
```

Operator Precedence

When an expression has multiple operators, then the evaluation of expression takes place in the order of operator precedence. The operators with higher precedence are evaluated first. Operator precedences of MySQL are shown in the following list, from highest precedence to the lowest. Operators that are shown together on a line have the same precedence.

```
!
-(unary minus), ~(unary bit inversion)
^
*,/,DIV,%,MOD
-,+
<,>
```

python :

&
|
==, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN BETWEEN, CASE, WHEN THEN, ELSE
NOT
&&, AND
XOR
||, OR
:=

Sorting Results ORDER BY clause

Whenever a SELECT query is executed, the resulting rows emerge in a redecided order. You can sort the results or a query in a specific order using ORDER BY clause. The ORDER BY clause allows sorting of query results by one or more columns. The sorting can be done either in ascending or descending order, the default order is ascending. The data in the table is not sorted ; only the results that appear on the screen are sorted. The ORDER BY (www.cbsepythonedu.com) clause is used as

```
SELECT < column name > [, < column name > ,...]  
FROM < table name >  
[ORDER BY < column name > ] ;
```

To specify the sort order, we may specify DESC for descending order or ASC for ascending order. (www.cbsepythonedu.com) Furthermore, ordering can be performed on multiple attributes.

MYSQL FUNCTIONS

A function is a special type of predefined command set that performs some operation and returns a single value. Functions operate on zero, one, two or more values that are provided to them. The values that are provided to functions are called parameters or arguments. The MySQL functions have been categorised into various categories, such as string functions. Mathematical functions, Date and Time functions and so on. Although MySQL offers a big bouquet of functions, yet in this discussion we shall remain limited to some (www.cbsepythonedu.com) common functions.

String Functions

The string function of MySQL can manipulate the text string in many ways. Some commonly used string functions are being discussed below.

Function	Description	Examples
CHAR ()	Returns the character for each integer passed	1.SELECT CHAR (70,65,67,69) 2.SELECT CHAR (65,67,3, '63.3)
CONCAT ()	Returns concatenated string	SELECT CONCAT (name, aggregate) as "Name Marks" From student Where age =14 or age =16 ;
LOWER () / LCASE ()	Returns the argument in lowercase	SELECT LOWER ('MR. OBAMA') AS "LowerName1" Lower ('MS. Gandhi') AS

python :

SUBSTRING ()	Returns the substring as specified	“LowerName2” 1.SELECT SUBSTR (‘ABCDEFGH’ 3,4) “Subs” 2.SELECT SUBSTR (‘ABCDEFGH’, -5,4) “Subs”
SUBSTR ()		SELECT UPPER (‘Large’) “Uppercase”.
UPPER ()/	Converts to uppercase	or SELECT UCASE (‘Large’) “Uppercase”
UCASE ()		SELECT LTRIM (‘ RDBMS MySQL ’);
LTRIM ()	Removes leading space	SELECT RTRIM (‘ RDBMS MySQL ’);
RTRIM ()	Removes trailing spaces	SELECT TRIM (‘ Bar One ’);
TRIM ()	Removes leading and Trailing spaces	[www.cbsepythonedu.com]
INSTR ()	Returns the index of the first occurrence of substring	SELECT INSTR (‘CORPORATE FLOOR’, ‘OR’) AS Instr; ;
LENGTH ()	Returns the length of a string in bytes	SELECT LENGTH (‘CANDIDE’) “Length in Character”
LEFT ()	Returns the leftmost number of characters as specified	SELECT LEFT (‘USS/23/67/09’, 3);
RIGHT ()	Returns the specified right most number of characters	[www.cbsepythonedu.com] SELECT RIGHT (‘USS/23/67/09’, 2);
MID()	Returns a substring starting from the specified position	SELECT SUBSTRING (‘Quadratically’, 5,6); or SELECT MID (‘Quadratically’, 5,6);

Numeric Function

The number function are those function that accept numeric values and after performing the required operation, return numeric values. Some useful numeric functions are being discussed below :

Fuction	Description	Example
MOD ()	Returns the remainder of one expression by dividing by another expression	SELECT MOD (11,4) “Modulus”
POWER () / POW ()	Returns the value of one expression raised to the power of another expression	SELECT POWER (3,2) “Raised” [www.cbsepythonedu.com]
ROUND ()	Returns numeric expression rounded to an integer. Can be used to round an expression to a number of decimal points	SELECT ROUND (15.193, 1) “Round” [www.cbsepythonedu.com]
SIGN ()	This function returns sign of a given number	SELECT SIGN (-15) “Sign”
SORT ()	Returns the non-negative square root of numeric expression	SELECT SQRT (26) “Square root”
TRUNCATE ()	Returns numeric exp1 truncated to exp2 decimal place. If exp2 is 0, “Truncate” then the result will have no decimal point	SELECT TRUNCATE (15.79, 1) [www.cbsepythonedu.com]

Date and Time Functions

Date functions operate on values of the DATE datatype

python :

Function	Description	Example
CURDATE ()/ CURRENT_DATE()	Returns the current date	SELECT CURDATE () [www.cbsepythonedu.com]
CURRENT_DATE DATE ()	Extracts the date part of a date or date-time expression	SELECT DATE ('2020-12-31-01-02-03')
MONTH ()	Returns the month from the date passed	SELECT MONTH ('2020-02-03') ; [www.cbsepythonedu.com]
YEAR ()	Returns the year	SELECT YEAR ('2020-02-03') ;
DAYNAME ()	Returns the name of the weekly	SELECT DAYNAME ('2020-02-03') ;
DAYOFMONTH()	Returns the dayof the month(1-31)	SELECT DAYOFMONTH ('2020-02-03') ;
DAYOFWEEK ()	Returns the weekly index of the argument	SELECT DAYOFWEEK ('2020-02-03') [www.cbsepythonedu.com]
DAYOFYEAR ()	Returns the day of the year (1-366)	SELECT DAYOFYEAR ('2020-02-13')
NOW ()	Returns the time at which the function executes	SELECT NOW () :
SYSDATE ()	Returns the current date and time	SELECT NOW (), SLEEP(2), NOW() ;