MySQL Basics

Here, you will learn how to use the basic form of the MySQL SELECT statement to query data from a table.

## **Introduction to MySQL SELECT statement**

The SELECT statement allows you to select data from one or more tables. To write a SELECT statement in MySQL, you use this syntax:

SELECT select\_list

FROM table\_name;

In this syntax:

* First, specify one or more columns from which you want to select data after the SELECT keyword. If the select\_list has multiple columns, you need to separate them by a comma (,).
* Second, specify the name of the table from which you want to select data after the FROM keyword.

The semicolon (;) is optional. It denotes the end of a statement. If you have two or more statements, you need to use the semicolon(;) to separate them so that MySQL will execute each statement individually.

The SELECT and FROM are the keywords. You can use either uppercase or lowercase because SQL is case-insensitive, you can write the SQL statement in lowercase, uppercase, etc. For example:

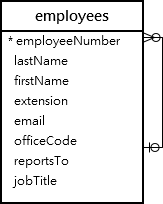
select select\_list

from table\_name;

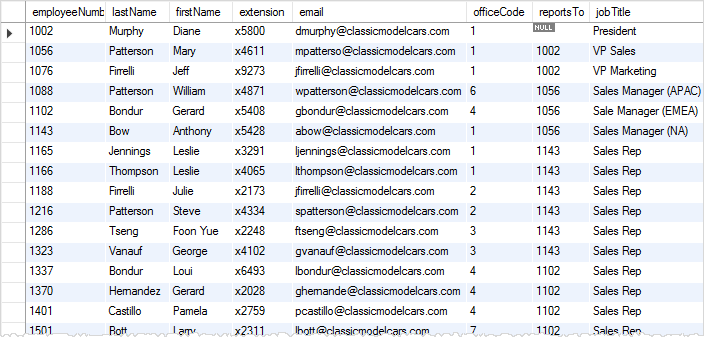
When executing the SELECT statement, MySQL evaluates the FROM clause before the SELECT clause:

## **MySQL SELECT statement examples**

We’ll use the employees table in the sample database for the following examples.



The employees table has eight columns: employeeNumber, lastName, firstName, extension, email, officeCode, reportsTo, and jobTitle. The table also has many rows as shown in the following picture:



### **A) Using the MySQL SELECT statement to retrieve data from a single column example**

The following example uses the SELECT statement to select the last names of all employees:

SELECT lastName

FROM employees;

Here’s the partial output:

+-----------+

| lastName |

+-----------+

| Murphy |

| Patterson |

| Firrelli |

| Patterson |

| Bondur |

| Bow |

| Jennings |

...

The result of a SELECT statement is called a result set as it’s a set of rows that results from the query.

### **B) Using the MySQL SELECT statement to query data from multiple columns example**

The following example uses the SELECT statement to get the first name, last name, and job title of employees:

SELECT

lastName,

firstName,

jobTitle

FROM

employees;

Even though the employees table has many columns, the SELECT statement returns data of three columns lastName, firstName, and jobTitle specified in the SELECT clause:

+-----------+-----------+----------------------+

| lastname | firstname | jobtitle |

+-----------+-----------+----------------------+

| Murphy | Diane | President |

| Patterson | Mary | VP Sales |

| Firrelli | Jeff | VP Marketing |

| Patterson | William | Sales Manager (APAC) |

| Bondur | Gerard | Sale Manager (EMEA) |

...

### **C) Using the MySQL SELECT statement to retrieve data from all columns example**

If you want to select data from all the columns of the employees table, you can specify all the column names in the SELECT clause like this:

SELECT employeeNumber,

lastName,

firstName,

extension,

email,

officeCode,

reportsTo,

jobTitle

FROM employees;

Alternatively, you can use the asterisk (\*) which is the shorthand for all columns. For example:

SELECT \*

FROM employees;

The query returns data from all the columns of the employees table.

The SELECT \* is often called “select star” or “select all” since it selects data from all columns of the table. In practice, you should use the SELECT \* for the ad-hoc queries only.

If you embed the SELECT statement in the code such as PHP, Java, Python, Node.js, you should explicitly specify the columns from which you want to select data.

## **Summary**

* Use the SELECT statement to select data from a table.
* Use the SELECT \* to select data from all columns of a table.

 Here you’ll learn how to use the MySQL SELECT statement without referencing any table.

Typically, you use a SELECT statement to select data from a table in the database:

SELECT select\_list

FROM table\_name;

MySQL doesn’t require the FROM clause. It means that you can have a SELECT statement without the FROM clause like this:

SELECT select\_list;

Here’s a trivial example:

SELECT 1 + 1;

Output:

+-------+

| 1 + 1 |

+-------+

| 2 |

+-------+

1 row in set (0.00 sec)

MySQL has many built-in functions like string, date, and Math functions. And you can use the SELECT statement to execute these functions.

The following example returns the current date and time of the MySQL server:

SELECT NOW();

Output:

+---------------------+

| NOW() |

+---------------------+

| 2021-07-26 08:08:02 |

+---------------------+

1 row in set (0.00 sec)

The NOW() function returns the current date & time of the server on which MySQL runs. The NOW() function doesn’t have any parameters. To call it, you place the parentheses () after the function name.

If a function has parameters, you need to pass arguments into it. For example, concatenate strings into one string, you can use the CONCAT() function:

SELECT CONCAT('John',' ','Doe');

Output:

+--------------------------+

| CONCAT('John',' ','Doe') |

+--------------------------+

| John Doe |

+--------------------------+

1 row in set (0.00 sec)

The CONCAT() function accepts one or more strings and concatenates them into a single string.

## **The dual table**

Sometimes, you still need to use the FROM clause but you don’t want to reference any actual table. In this case, you can use the dual table in the FROM clause:

SELECT select\_list

FROM dual;

The dual table is a dummy table, not an actual table.

The dual is also necessary in case you need other clauses of the SELECT statement. Without the FROM clause, the SELECT statement would not be valid.

select now() from dual;

## **A quick introduction to the column alias**

By default, MySQL uses the expression specified in the SELECT clause as the column name of the result set. To change a column name of the result set, you can use a column alias:

SELECT expression AS column\_alias;

To assign an alias to a column, you place the AS keyword after the expression followed by a column alias. The AS keyword is optional, so you can skip it like this:

SELECT expression column\_alias;

For example:

SELECT CONCAT('John',' ','Doe') AS name;

Output:

+----------+

| name |

+----------+

| John Doe |

+----------+

1 row in set (0.00 sec)

If the column alias contains spaces, you need to place it inside quotes like this:

SELECT CONCAT('Jane',' ','Doe') AS 'Full name';

Output:

+-----------+

| Full name |

+-----------+

| John Doe |

+-----------+

1 row in set (0.00 sec)

## **Summary**

* MySQL SELECT statement doesn’t require the FROM clause
* Use the dual table if you want to use the FROM clause but don’t want to reference a table.
* Assign an alias to a column to make it more readable.