SQL Tutorial

« W3Schools Home Next Chapter »

SQL is a standard language for accessing databases.

Our SQL tutorial will teach you how to use SQL to access and manipulate data in: MySQL, SQL Server, Access, Oracle, Sybase, DB2, and other database systems.

Examples in Each Chapter

With our online SQL editor, you can edit the SQL statements, and click on a button to view the result.

Example

```
SELECT * FROM Customers;
```

Try it yourself »

Click on the "Try it yourself" button to see how it works.

Start learning SQL now!

SQL Quiz Test

Test your SQL skills at W3Schools!

Start SQL Quiz!

SQL Quick Reference

An SQL Quick Reference. Print it and put it in your pocket.

SQL Quick Reference

SQL Data Types

Data types and ranges for Microsoft Access, MySQL and SQL Server.

SQL Data Types

W3Schools Exam

W3Schools' Online Certification

The perfect solution for professionals who need to balance work, family, and career building.

More than 10 000 certificates already issued!

Get Your Certificate »

The HTML Certificate documents your knowledge of HTML.

The HTML5 Certificate documents your knowledge of advanced HTML5.

The CSS Certificate documents your knowledge of advanced CSS.

The JavaScript Certificate documents your knowledge of JavaScript and HTML DOM.

The jQuery Certificate documents your knowledge of jQuery.

The PHP Certificate documents your knowledge of PHP and SQL (MySQL).

The XML Certificate documents your knowledge of XML, XML DOM and XSLT.

The Bootstrap Certificate documents your knowledge of the Bootstrap framework.

« W3Schools Home Next Chapter »

Introduction to SQL

« Previous Next Chapter »

SQL is a standard language for accessing and manipulating databases.

What is SQL?

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL is an ANSI (American National Standards Institute) standard

What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

SQL is a Standard - BUT....

Although SQL is an ANSI (American National Standards Institute) standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

Note: Most of the SQL database programs also have their own proprietary extensions in addition to the SQL standard!

Using SQL in Your Web Site

To build a web site that shows data from a database, you will need:

- An RDBMS database program (i.e. MS Access, SQL Server, MySQL)
- To use a server-side scripting language, like PHP or ASP
- To use SQL to get the data you want
- To use HTML / CSS

RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

The data in RDBMS is stored in database objects called tables.

A table is a collection of related data entries and it consists of columns and rows.

« Previous Next Chapter »

SQL Syntax

« Previous Next Chapter »

Database Tables

A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data.

In this tutorial we will use the well-known Northwind sample database (included in MS Access and MS SQL Server).

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

The table above contains five records (one for each customer) and seven columns (CustomerID, CustomerName, ContactName, Address, City, PostalCode, and Country).

SQL Statements

Most of the actions you need to perform on a database are done with SQL statements.

The following SQL statement selects all the records in the "Customers" table:

Example

```
SELECT * FROM Customers;
```

Try it yourself »

In this tutorial we will teach you all about the different SQL statements.

Keep in Mind That...

- SQL is NOT case sensitive: select is the same as SELECT

In this tutorial we will write all SQL keywords in upper-case.

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

In this tutorial, we will use semicolon at the end of each SQL statement.

Some of The Most Important SQL Commands

- **SELECT** - extracts data from a database
- **UPDATE** - updates data in a database
- **DELETE** - deletes data from a database

- **INSERT INTO** - inserts new data into a database
- **CREATE DATABASE** - creates a new database
- **ALTER DATABASE** - modifies a database
- **CREATE TABLE** - creates a new table
- **ALTER TABLE** - modifies a table
- **DROP TABLE** - deletes a table
- **CREATE INDEX** - creates an index (search key)
- **DROP INDEX** - deletes an index

« Previous Next Chapter »

SQL SELECT Statement

« Previous Next Chapter »

The SELECT statement is used to select data from a database.

The SQL SELECT Statement

The SELECT statement is used to select data from a database.

The result is stored in a result table, called the result-set.

SQL SELECT Syntax

```
SELECT column_name,column_name
FROM table_name;
```

and

```
SELECT * FROM table_name;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220

2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

SELECT Column Example

The following SQL statement selects the "CustomerName" and "City" columns from the "Customers" table:

Example

```
SELECT CustomerName,City FROM Customers;
```

Try it yourself »

SELECT * Example

The following SQL statement selects all the columns from the "Customers" table:

Example

```
SELECT * FROM Customers;
```

Try it yourself »

Navigation in a Result-set

Most database software systems allow navigation in the result-set with programming functions, like: Move-To-First-Record, Get-Record-Content, Move-To-Next-Record, etc.

Programming functions like these are not a part of this tutorial. To learn about accessing data with function calls, please visit our ASP tutorial or our PHP tutorial.

« Previous Next Chapter »

SQL SELECT DISTINCT Statement

« Previous Next Chapter »

The SELECT DISTINCT statement is used to return only distinct (different) values.

The SQL SELECT DISTINCT Statement

In a table, a column may contain many duplicate values; and sometimes you only want to list the different (distinct) values.

The DISTINCT keyword can be used to return only distinct (different) values.

SQL SELECT DISTINCT Syntax

```
SELECT DISTINCT column_name,column_name
FROM table_name;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502

3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

SELECT DISTINCT Example

The following SQL statement selects only the distinct values from the "City" columns from the "Customers" table:

Example

```
SELECT DISTINCT City FROM Customers;
```

Try it yourself »

« Previous Next Chapter »

SQL WHERE Clause

« Previous   Next Chapter »

The WHERE clause is used to filter records.

The SQL WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

SQL WHERE Syntax

```
SELECT column_name,column_name
FROM table_name
WHERE column_name operator value;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	**CustomerName**	**ContactName**	**Address**	**City**	**Post**
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502

4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

WHERE Clause Example

The following SQL statement selects all the customers from the country "Mexico", in the "Customers" table:

Example

```
SELECT * FROM Customers
WHERE Country='Mexico';
```

Try it yourself »

Text Fields vs. Numeric Fields

SQL requires single quotes around text values (most database systems will also allow double quotes).

However, numeric fields should not be enclosed in quotes:

Example

```
SELECT * FROM Customers
WHERE CustomerID=1;
```

Try it yourself »

Operators in The WHERE Clause

The following operators can be used in the WHERE clause:

Operator	**Description**
=	Equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

« Previous Next Chapter »

SQL AND & OR Operators

« Previous Next Chapter »

The AND & OR operators are used to filter records based on more than one condition.

The SQL AND & OR Operators

The AND operator displays a record if both the first condition AND the second condition are true.

The OR operator displays a record if either the first condition OR the second condition is true.

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1

5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958

AND Operator Example

The following SQL statement selects all customers from the country "Germany" AND the city "Berlin", in the "Customers" table:

Example

```
SELECT * FROM Customers
WHERE Country='Germany'
AND City='Berlin';
```

Try it yourself »

OR Operator Example

The following SQL statement selects all customers from the city "Berlin" OR "München", in the "Customers" table:

Example

```
SELECT * FROM Customers
WHERE City='Berlin'
OR City='München';
```

Try it yourself »

Combining AND & OR

You can also combine AND and OR (use parenthesis to form complex expressions).

The following SQL statement selects all customers from the country "Germany" AND the city must be equal to "Berlin" OR "München", in the "Customers" table:

Example

```
SELECT * FROM Customers
WHERE Country='Germany'
AND (City='Berlin' OR City='München');
```

Try it yourself »

« Previous Next Chapter »

SQL ORDER BY Keyword

« Previous Next Chapter »

The ORDER BY keyword is used to sort the result-set.

The SQL ORDER BY Keyword

The ORDER BY keyword is used to sort the result-set by one or more columns.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in a descending order, you can use the DESC keyword.

SQL ORDER BY Syntax

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC|DESC, column_name ASC|DESC;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502

3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

ORDER BY Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

Example

```
SELECT * FROM Customers
ORDER BY Country;
```

Try it yourself »

ORDER BY DESC Example

The following SQL statement selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column:

Example

```
SELECT * FROM Customers
ORDER BY Country DESC;
```

Try it yourself »

ORDER BY Several Columns Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column:

Example

```
SELECT * FROM Customers
ORDER BY Country, CustomerName;
```

Try it yourself »

ORDER BY Several Columns Example 2

The following SQL statement selects all customers from the "Customers" table, sorted ascending by the "Country" and descending by the "CustomerName" column:

Example

```
SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

Try it yourself »

« Previous Next Chapter »

SQL INSERT INTO Statement

« Previous Next Chapter »

The INSERT INTO statement is used to insert new records in a table.

The SQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

SQL INSERT INTO Syntax

It is possible to write the INSERT INTO statement in two forms.

The first form does not specify the column names where the data will be inserted, only their values:

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

The second form specifies both the column names and the values to be inserted:

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Posta
87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	90110
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	08737
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-01

INSERT INTO Example

Assume we wish to insert a new row in the "Customers" table.

We can use the following SQL statement:

Example

```
INSERT INTO Customers (CustomerName, ContactName, Address, City,
PostalCode, Country)
VALUES ('Cardinal','Tom B. Erichsen','Skagen
21','Stavanger','4006','Norway');
```

Try it yourself »

The selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	Post

87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	9011
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	0873
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	9812
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	2124
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-0
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006

Did you notice that we did not insert any number into the CustomerID field?
The CustomerID column is automatically updated with a unique number for each record in the table.

Insert Data Only in Specified Columns

It is also possible to only insert data in specific columns.

The following SQL statement will insert a new row, but only insert data in the "CustomerName", "City", and "Country" columns (and the CustomerID field will of course also be updated automatically):

Example

```
INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Cardinal', 'Stavanger', 'Norway');
```

Try it yourself »

The selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	Post
87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	9011
88	Wellington Importadora	Paula Parente	Rua do Mercado, 12	Resende	0873
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	9812
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	2124
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-0
92	Cardinal	null	null	Stavanger	null

« Previous Next Chapter »

SQL UPDATE Statement

« Previous Next Chapter »

The UPDATE statement is used to update records in a table.

The SQL UPDATE Statement

The UPDATE statement is used to update existing records in a table.

SQL UPDATE Syntax

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

Notice the WHERE clause in the SQL UPDATE statement!
The WHERE clause specifies which record or records that should be updated. If you omit the WHERE clause, all records will be updated!

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y	Ana Trujillo	Avda. de la Constitución	México D.F.	0502

	helados		2222		
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

SQL UPDATE Example

Assume we wish to update the customer "Alfreds Futterkiste" with a new contact person and city.

We use the following SQL statement:

Example

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Hamburg'
WHERE CustomerName='Alfreds Futterkiste';
```

Try it yourself »

The selection from the "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	Pos
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Hamburg	122
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	050
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	050

4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-9

Update Warning!

Be careful when updating records. If we had omitted the WHERE clause, in the example above, like this:

```
UPDATE Customers
SET ContactName='Alfred Schmidt', City='Hamburg';
```

The "Customers" table would have looked like this:

CustomerID	CustomerName	ContactName	Address	City	Pos
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Hamburg	122
2	Ana Trujillo Emparedados y helados	Alfred Schmidt	Avda. de la Constitución 2222	Hamburg	050
3	Antonio Moreno Taquería	Alfred Schmidt	Mataderos 2312	Hamburg	050
4	Around the Horn	Alfred Schmidt	120 Hanover Sq.	Hamburg	WA
5	Berglunds snabbköp	Alfred Schmidt	Berguvsvägen 8	Hamburg	S-9

« Previous Next Chapter »

SQL DELETE Statement

« Previous Next Chapter »

The DELETE statement is used to delete records in a table.

The SQL DELETE Statement

The DELETE statement is used to delete rows in a table.

SQL DELETE Syntax

```
DELETE FROM table_name
WHERE some_column=some_value;
```

Notice the WHERE clause in the SQL DELETE statement!
The WHERE clause specifies which record or records that should be deleted. If you omit the WHERE clause, all records will be deleted!

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502

3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

SQL DELETE Example

Assume we wish to delete the customer "Alfreds Futterkiste" from the "Customers" table.

We use the following SQL statement:

Example

```
DELETE FROM Customers
WHERE CustomerName='Alfreds Futterkiste' AND ContactName='Maria
Anders';
```

Try it yourself »

The "Customers" table will now look like this:

CustomerID	CustomerName	ContactName	Address	City	Post
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1

5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958

Delete All Data

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

```
DELETE FROM table_name;

or

DELETE * FROM table_name;
```

Note: Be very careful when deleting records. You cannot undo this statement!

« Previous Next Chapter »

SQL Injection

« Previous Next Chapter »

An SQL Injection can destroy your database.

SQL in Web Pages

In the previous chapters, you have learned to retrieve (and update) database data, using SQL.

When SQL is used to display data on a web page, it is common to let web users input their own search values.

Since SQL statements are text only, it is easy, with a little piece of computer code, to dynamically change SQL statements to provide the user with selected data:

Server Code

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

The example above, creates a select statement by adding a variable (txtUserId) to a select string. The variable is fetched from the user input (Request) to the page.

The rest of this chapter describes the potential dangers of using user input in SQL statements.

SQL Injection

SQL injection is a technique where malicious users can inject SQL commands into an SQL statement, via web page input.

Injected SQL commands can alter SQL statement and compromise the security of a web application.

SQL Injection Based on 1=1 is Always True

Look at the example above, one more time.

Let's say that the original purpose of the code was to create an SQL statement to select a user with a given user id.

If there is nothing to prevent a user from entering "wrong" input, the user can enter some "smart" input like this:

UserId:

105 or 1=1

Server Result

```
SELECT * FROM Users WHERE UserId = 105 or 1=1
```

The SQL above is valid. It will return all rows from the table Users, since **WHERE 1=1** is always true.

Does the example above seem dangerous? What if the Users table contains names and passwords?

The SQL statement above is much the same as this:

```
SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1
```

A smart hacker might get access to all the user names and passwords in a database by simply inserting 105 or 1=1 into the input box.

SQL Injection Based on ""="" is Always True

Here is a common construction, used to verify user login to a web site:

User Name:

Password:

Server Code

```
uName = getRequestString("UserName");
uPass = getRequestString("UserPass");

sql = "SELECT * FROM Users WHERE Name ='" + uName + "' AND Pass ='"
+ uPass + "'"
```

A smart hacker might get access to user names and passwords in a database by simply inserting " or ""=" into the user name or password text box.

The code at the server will create a valid SQL statement like this:

Result

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

The result SQL is valid. It will return all rows from the table Users, since **WHERE ""=""** is always true.

SQL Injection Based on Batched SQL Statements

Most databases support batched SQL statement, separated by semicolon.

Example

```
SELECT * FROM Users; DROP TABLE Suppliers
```

The SQL above will return all rows in the Users table, and then delete the table called Suppliers.

If we had the following server code:

Server Code

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

And the following input:

User id:
105; DROP TABLE Suppliers

The code at the server would create a valid SQL statement like this:

Result

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers
```

Parameters for Protection

Some web developers use a "blacklist" of words or characters to search for in SQL input, to prevent SQL injection attacks.

This is not a very good idea. Many of these words (like delete or drop) and characters (like semicolons and quotation marks), are used in common language, and should be allowed in many types of input.

(In fact it should be perfectly legal to input an SQL statement in a database field.)

The only proven way to protect a web site from SQL injection attacks, is to use SQL parameters.

SQL parameters are values that are added to an SQL query at execution time, in a controlled manner.

ASP.NET Razor Example

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = @0";
db.Execute(txtSQL,txtUserId);
```

Note that parameters are represented in the SQL statement by a @ marker.

The SQL engine checks each parameter to ensure that it is correct for its column and are treated literally, and not as part of the SQL to be executed.

Another Example

```
txtNam = getRequestString("CustomerName");
txtAdd = getRequestString("Address");
txtCit = getRequestString("City");
txtSQL = "INSERT INTO Customers (CustomerName,Address,City)
Values(@0,@1,@2)";
db.Execute(txtSQL,txtNam,txtAdd,txtCit);
```

You have just learned to avoid SQL injection. One of the top website vulnerabilities.

Examples

The following examples shows how to build parameterized queries in some common web languages.

SELECT STATEMENT IN ASP.NET:

```
txtUserId = getRequestString("UserId");
sql = "SELECT * FROM Customers WHERE CustomerId = @0";
command = new SqlCommand(sql);
command.Parameters.AddWithValue("@0",txtUserID);
command.ExecuteReader();
```

INSERT INTO STATEMENT IN ASP.NET:

```
txtNam = getRequestString("CustomerName");
txtAdd = getRequestString("Address");
txtCit = getRequestString("City");
txtSQL = "INSERT INTO Customers (CustomerName,Address,City)
Values(@0,@1,@2)";
command = new SqlCommand(txtSQL);
command.Parameters.AddWithValue("@0",txtNam);
command.Parameters.AddWithValue("@1",txtAdd);
command.Parameters.AddWithValue("@2",txtCit);
command.ExecuteNonQuery();
```

INSERT INTO STATEMENT IN PHP:

```
$stmt = $dbh->prepare("INSERT INTO Customers
(CustomerName,Address,City)
VALUES (:nam, :add, :cit)");
$stmt->bindParam(':nam', $txtNam);
$stmt->bindParam(':add', $txtAdd);
$stmt->bindParam(':cit', $txtCit);
$stmt->execute();
```

« Previous Next Chapter »

SQL SELECT TOP Clause

« Previous Next Chapter »

The SQL SELECT TOP Clause

The SELECT TOP clause is used to specify the number of records to return.

The SELECT TOP clause can be very useful on large tables with thousands of records. Returning a large number of records can impact on performance.

Note: Not all database systems support the SELECT TOP clause.

SQL Server / MS Access Syntax

```
SELECT TOP number|percent column_name(s)
FROM table_name;
```

SQL SELECT TOP Equivalent in MySQL and Oracle

MySQL Syntax

```
SELECT column_name(s)
FROM table_name
LIMIT number;
```

Example

```
SELECT *
FROM Persons
LIMIT 5;
```

Oracle Syntax

```
SELECT column_name(s)
FROM table_name
WHERE ROWNUM <= number;
```

Example

```
SELECT *
FROM Persons
WHERE ROWNUM <=5;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

SQL SELECT TOP Example

The following SQL statement selects the two first records from the "Customers" table:

Example

```
SELECT TOP 2 * FROM Customers;
```

Try it yourself »

SQL SELECT TOP PERCENT Example

The following SQL statement selects the first 50% of the records from the "Customers" table:

Example

```
SELECT TOP 50 PERCENT * FROM Customers;
```

Try it yourself »

« Previous Next Chapter »

SQL LIKE Operator

« Previous Next Chapter »

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

The SQL LIKE Operator

The LIKE operator is used to search for a specified pattern in a column.

SQL LIKE Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno	Antonio	Mataderos	México	0502

	Taquería	Moreno	2312	D.F.	
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

SQL LIKE Operator Examples

The following SQL statement selects all customers with a City starting with the letter "s":

Example

```
SELECT * FROM Customers
WHERE City LIKE 's%';
```

Try it yourself »

Tip: The "%" sign is used to define wildcards (missing letters) both before and after the pattern. You will learn more about wildcards in the next chapter.

The following SQL statement selects all customers with a City ending with the letter "s":

Example

```
SELECT * FROM Customers
WHERE City LIKE '%s';
```

Try it yourself »

The following SQL statement selects all customers with a Country containing the pattern "land":

Example

```
SELECT * FROM Customers
WHERE Country LIKE '%land%';
```

Try it yourself »

Using the NOT keyword allows you to select records that do NOT match the pattern.

The following SQL statement selects all customers with Country NOT containing the pattern "land":

Example

```
SELECT * FROM Customers
WHERE Country NOT LIKE '%land%';
```

Try it yourself »

« Previous Next Chapter »

SQL Wildcards

« Previous   Next Chapter »

A wildcard character can be used to substitute for any other character(s) in a string.

SQL Wildcard Characters

In SQL, wildcard characters are used with the SQL LIKE operator.

SQL wildcards are used to search for data within a table.

With SQL, the wildcards are:

Wildcard	**Description**
%	A substitute for zero or more characters
_	A substitute for a single character
[*charlist*]	Sets and ranges of characters to match
[^*charlist*] or [!*charlist*]	Matches only a character NOT specified within the brackets

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	**CustomerName**	**ContactName**	**Address**	**City**	**Post**

1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

Using the SQL % Wildcard

The following SQL statement selects all customers with a City starting with "ber":

Example

```
SELECT * FROM Customers
WHERE City LIKE 'ber%';
```

Try it yourself »

The following SQL statement selects all customers with a City containing the pattern "es":

Example

```
SELECT * FROM Customers
WHERE City LIKE '%es%';
```

Try it yourself »

Using the SQL _ Wildcard

The following SQL statement selects all customers with a City starting with any character, followed by "erlin":

Example

```
SELECT * FROM Customers
WHERE City LIKE '_erlin';
```

Try it yourself »

The following SQL statement selects all customers with a City starting with "L", followed by any character, followed by "n", followed by any character, followed by "on":

Example

```
SELECT * FROM Customers
WHERE City LIKE 'L_n_on';
```

Try it yourself »

Using the SQL [charlist] Wildcard

The following SQL statement selects all customers with a City starting with "b", "s", or "p":

Example

```
SELECT * FROM Customers
WHERE City LIKE '[bsp]%';
```

Try it yourself »

The following SQL statement selects all customers with a City starting with "a", "b", or "c":

Example

```
SELECT * FROM Customers
WHERE City LIKE '[a-c]%';
```

Try it yourself »

The following SQL statement selects all customers with a City NOT starting with "b", "s", or "p":

Example

```
SELECT * FROM Customers
WHERE City LIKE '[!bsp]%';

or

SELECT * FROM Customers
WHERE City NOT LIKE '[bsp]%';
```

Try it yourself »

« Previous Next Chapter »

SQL IN Operator

« Previous Next Chapter »

The IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

SQL IN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...);
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Post
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	1220
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	0502
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	0502
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1

5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-95

IN Operator Example

The following SQL statement selects all customers with a City of "Paris" or "London":

Example

```
SELECT * FROM Customers
WHERE City IN ('Paris','London');
```

Try it yourself »

« Previous Next Chapter »

SQL BETWEEN Operator

« Previous Next Chapter »

The BETWEEN operator is used to select values within a range.

The SQL BETWEEN Operator

The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

SQL BETWEEN Syntax

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Products" table:

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19

3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	1	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	1	2	36 boxes	21.35

BETWEEN Operator Example

The following SQL statement selects all products with a price BETWEEN 10 and 20:

Example

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;
```

Try it yourself »

NOT BETWEEN Operator Example

To display the products outside the range of the previous example, use NOT BETWEEN:

Example

```
SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;
```

Try it yourself »

BETWEEN Operator with IN Example

The following SQL statement selects all products with a price BETWEEN 10 and 20, but products with a CategoryID of 1,2, or 3 should not be displayed:

Example

```
SELECT * FROM Products
WHERE (Price BETWEEN 10 AND 20)
AND NOT CategoryID IN (1,2,3);
```

Try it yourself »

BETWEEN Operator with Text Value Example

The following SQL statement selects all products with a ProductName beginning with any of the letter BETWEEN 'C' and 'M':

Example

```
SELECT * FROM Products
WHERE ProductName BETWEEN 'C' AND 'M';
```

Try it yourself »

NOT BETWEEN Operator with Text Value Example

The following SQL statement selects all products with a ProductName beginning with any of the letter NOT BETWEEN 'C' and 'M':

Example

```
SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'C' AND 'M';
```

Try it yourself »

Sample Table

Below is a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10248	90	5	7/4/1996	3
10249	81	6	7/5/1996	1
10250	34	4	7/8/1996	2
10251	84	3	7/9/1996	1
10252	76	4	7/10/1996	2

BETWEEN Operator with Date Value Example

The following SQL statement selects all orders with an OrderDate BETWEEN '04-July-1996' and '09-July-1996':

Example

```
SELECT * FROM Orders
WHERE OrderDate BETWEEN #07/04/1996# AND #07/09/1996#;
```

Try it yourself »

Notice that the BETWEEN operator can produce different result in different databases!
In some databases, BETWEEN selects fields that are between and excluding the test values.
In other databases, BETWEEN selects fields that are between and including the test values.
And in other databases, BETWEEN selects fields between the test values, including the first test value and excluding the last test value.

Therefore: Check how your database treats the BETWEEN operator!

« Previous Next Chapter »

SQL Aliases

« Previous Next Chapter »

SQL aliases are used to temporarily rename a table or a column heading.

SQL Aliases

SQL aliases are used to give a database table, or a column in a table, a temporary name.

Basically aliases are created to make column names more readable.

SQL Alias Syntax for Columns

```
SELECT column_name AS alias_name
FROM table_name;
```

SQL Alias Syntax for Tables

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal
2	Ana Trujillo	Ana Trujillo	Avda. de la	México	05021

	Emparedados y helados		Constitución 2222	D.F.	
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1

And a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10354	58	8	1996-11-14	3
10355	4	6	1996-11-15	1
10356	86	6	1996-11-18	2

Alias Example for Table Columns

The following SQL statement specifies two aliases, one for the CustomerName column and one for the ContactName column. **Tip:** It requires double quotation marks or square brackets if the column name contains spaces:

Example

```
SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;
```

Try it yourself »

In the following SQL statement we combine four columns (Address, City, PostalCode, and Country) and create an alias named "Address":

Example

```
SELECT CustomerName, Address+', '+City+', '+PostalCode+', '+Country
AS Address
FROM Customers;
```

Try it yourself »

Note: To get the SQL statement above to work in MySQL use the following:

```
SELECT CustomerName, CONCAT(Address,', ',City,', ',PostalCode,',
',Country) AS Address
FROM Customers;
```

Alias Example for Tables

The following SQL statement selects all the orders from the customer with CustomerID=4 (Around the Horn). We use the "Customers" and "Orders" tables, and give them the table aliases of "c" and "o" respectively (Here we have used aliases to make the SQL shorter):

Example

```
SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName="Around the Horn" AND
c.CustomerID=o.CustomerID;
```

Try it yourself »

The same SQL statement without aliases:

Example

```
SELECT Orders.OrderID, Orders.OrderDate, Customers.CustomerName
FROM Customers, Orders
WHERE Customers.CustomerName="Around the Horn" AND
Customers.CustomerID=Orders.CustomerID;
```

Try it yourself »

Aliases can be useful when:

- There are more than one table involved in a query
- Functions are used in the query
- Column names are big or not very readable
- Two or more columns are combined together

« Previous Next Chapter »

SQL Joins

« Previous Next Chapter »

SQL joins are used to combine rows from two or more tables.

SQL JOIN

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

The most common type of join is: **SQL INNER JOIN (simple join)**. An SQL INNER JOIN returns all rows from multiple tables where the join condition is met.

Let's look at a selection from the "Orders" table:

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

Then, have a look at a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

Notice that the "CustomerID" column in the "Orders" table refers to the "CustomerID" in the "Customers" table. The relationship between the two tables above is the "CustomerID" column.

Then, if we run the following SQL statement (that contains an INNER JOIN):

Example

```
SELECT Orders.OrderID, Customers.CustomerName, Orders.OrderDate
FROM Orders
INNER JOIN Customers
ON Orders.CustomerID=Customers.CustomerID;
```

Try it yourself »

it will produce something like this:

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

Different SQL JOINs

Before we continue with examples, we will list the types of the different SQL JOINs you can use:

- **INNER JOIN**: Returns all rows when there is at least one match in BOTH tables
- **LEFT JOIN**: Return all rows from the left table, and the matched rows from the right table

- **RIGHT JOIN**: Return all rows from the right table, and the matched rows from the left table
- **FULL JOIN**: Return all rows when there is a match in ONE of the tables

« Previous Next Chapter »

SQL INNER JOIN Keyword

« Previous Next Chapter »

SQL INNER JOIN Keyword

The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.

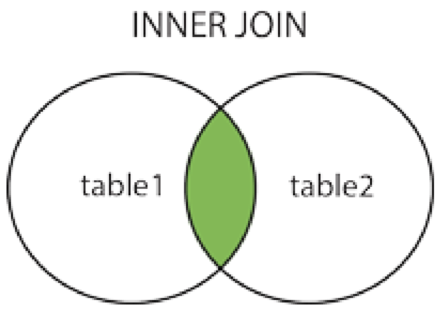
SQL INNER JOIN Syntax

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name=table2.column_name;
```

or:

```
SELECT column_name(s)
FROM table1
JOIN table2
ON table1.column_name=table2.column_name;
```

PS! INNER JOIN is the same as JOIN.



Demo Database

In this tutorial we will use the well-known Northwind sample database.

Below is a selection from the "Customers" table:

CustomerID	CustomerName	ContactName	Address	City	Postal
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023

And a selection from the "Orders" table:

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10308	2	7	1996-09-18	3
10309	37	3	1996-09-19	1
10310	77	8	1996-09-20	2

SQL INNER JOIN Example

The following SQL statement will return all customers with orders:

Example

```
SELECT Customers.CustomerName, Orders.OrderID
FROM Customers
INNER JOIN Orders
```

```
ON Customers.CustomerID=Orders.CustomerID
ORDER BY Customers.CustomerName;
```

Try it yourself »

Note: The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns. If there are rows in the "Customers" table that do not have matches in "Orders", these customers will NOT be listed.

« Previous Next Chapter »