# Chapter Five

# Introduction to SQL

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 many 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!

## RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems like 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 collections of related data entries and it consists of columns and rows.

# SQL Syntax

## 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.

Below is an example of a table called "Persons":

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

The table above contains three records (one for each person) and five columns (P\_Id, LastName, FirstName, Address, and City).

## SQL Statements

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

The following SQL statement will select all the records in the "Persons" table:

|  |
| --- |
| SELECT \* FROM Persons |

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

## Keep in Mind That...

* SQL is not case sensitive

## 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.

We are using MS Access and SQL Server 2000 and we do not have to put a semicolon after each SQL statement, but some database programs force you to use it.

**SQL DML and DDL**

SQL can be divided into two parts: The Data Manipulation Language (DML) and the Data Definition Language (DDL).

The DDL part of SQL permits database tables to be created or deleted. It also define indexes (keys), specify links between tables, and impose constraints between tables. The most important DDL statements in SQL are:

* **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

The query and update commands form the DML part of SQL:

* **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

**An SQL SELECT Example**

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen |  |  |  |

Now we want to select the content of the columns named "LastName" and "FirstName" from the table above.

We use the following SELECT statement:

|  |
| --- |
| SELECT LastName,FirstName FROM Persons |

The result-set will look like this:

|  |  |
| --- | --- |
| **LastName** | **FirstName** |
| Hansen | Ola |
| Svendson | Tove |
| Pettersen | Kari |

**SELECT \* Example**

Now we want to select all the columns from the "Persons" table.

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons |

**Tip:** The asterisk (\*) is a quick way of selecting all columns!

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

## The SQL SELECT DISTINCT Statement

In a table, some of the columns may contain duplicate values. This is not a problem, however, sometimes you will want to list only the different (distinct) values in a table.

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

### SQL SELECT DISTINCT Syntax

|  |
| --- |
| SELECT DISTINCT column\_name(s)  FROM table\_name |

## SELECT DISTINCT Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

Now we want to select only the distinct values from the column named "City" from the table above.

We use the following SELECT statement:

|  |
| --- |
| SELECT DISTINCT City FROM Persons |

The result-set will look like this:

|  |
| --- |
| **City** |
| Sandnes |
| Stavanger |

# SQL WHERE Clause

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

### SQL WHERE Syntax

|  |
| --- |
| SELECT column\_name(s)  FROM table\_name  WHERE column\_name operator value |

## WHERE Clause Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

Now we want to select only the persons living in the city "Sandnes" from the table above.

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons  WHERE City='Sandnes' |

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

## Quotes Around Text Fields

SQL uses single quotes around text values (most database systems will also accept double quotes).

Although, numeric values should not be enclosed in quotes.

For text values:

|  |
| --- |
| This is correct:  SELECT \* FROM Persons WHERE FirstName='Tove'  This is wrong:  SELECT \* FROM Persons WHERE FirstName=Tove |

For numeric values:

|  |
| --- |
| This is correct:  SELECT \* FROM Persons WHERE Year=1965  This is wrong:  SELECT \* FROM Persons WHERE Year='1965' |

## Operators Allowed in the WHERE Clause

With the WHERE clause, the following operators can be used:

|  |  |
| --- | --- |
| **Operator** | **Description** |
| = | Equal |
| <> | Not equal |
| > | Greater than |
| < | Less than |
| >= | Greater than or equal |
| <= | Less than or equal |
| BETWEEN | Between an inclusive range |
| LIKE | Search for a pattern |
| IN | If you know the exact value you want to return for at least one of the columns |

**Note:** In some versions of SQL the <> operator may be written as !=

# SQL AND & OR Operators

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

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

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

## AND Operator Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

Now we want to select only the persons with the first name equal to "Tove" AND the last name equal to "Svendson":

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons  WHERE FirstName='Tove'  AND LastName='Svendson' |

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

## OR Operator Example

Now we want to select only the persons with the first name equal to "Tove" OR the first name equal to "Ola":

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons  WHERE FirstName='Tove'  OR FirstName='Ola' |

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

## Combining AND & OR

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

Now we want to select only the persons with the last name equal to "Svendson" AND the first name equal to "Tove" OR to "Ola":

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons WHERE  LastName='Svendson'  AND (FirstName='Tove' OR FirstName='Ola') |

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

**SQL ORDER BY Keyword**

The ORDER BY keyword is used to sort the result-set by a specified column.

The ORDER BY keyword sort the records in ascending order by default.

If you want to sort the records in a descending order, you can use the DESC keyword.

### SQL ORDER BY Syntax

|  |
| --- |
| SELECT column\_name(s)  FROM table\_name  ORDER BY column\_name(s) ASC|DESC |

## ORDER BY Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Tom | Vingvn 23 | Stavanger |

Now we want to select all the persons from the table above, however, we want to sort the persons by their last name.

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons  ORDER BY LastName |

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 4 | Nilsen | Tom | Vingvn 23 | Stavanger |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |

## ORDER BY DESC Example

Now we want to select all the persons from the table above, however, we want to sort the persons descending by their last name.

We use the following SELECT statement:

|  |
| --- |
| SELECT \* FROM Persons  ORDER BY LastName DESC |

The result-set will look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Tom | Vingvn 23 | Stavanger |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |

# SQL INSERT INTO Statement

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

### SQL INSERT INTO Syntax

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

The first form doesn't 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,...) |

## SQL INSERT INTO Example

We have the following "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |

Now we want to insert a new row in the "Persons" table.

We use the following SQL statement:

|  |
| --- |
| INSERT INTO Persons  VALUES (4,'Nilsen', 'Johan', 'Bakken 2', 'Stavanger') |

The "Persons" table will now look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Johan | Bakken 2 | Stavanger |

## Insert Data Only in Specified Columns

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

The following SQL statement will add a new row, but only add data in the "P\_Id", "LastName" and the "FirstName" columns:

|  |
| --- |
| INSERT INTO Persons (P\_Id, LastName, FirstName)  VALUES (5, 'Tjessem', 'Jakob') |

The "Persons" table will now look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Johan | Bakken 2 | Stavanger |
| 5 | Tjessem | Jakob |  |  |

# SQL UPDATE Statement

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

### SQL UPDATE Syntax

|  |
| --- |
| UPDATE table\_name  SET column1=value, column2=value2,...  WHERE some\_column=some\_value |

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

## SQL UPDATE Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Johan | Bakken 2 | Stavanger |
| 5 | Tjessem | Jakob |  |  |

Now we want to update the person "Tjessem, Jakob" in the "Persons" table.

We use the following SQL statement:

|  |
| --- |
| UPDATE Persons  SET Address='Nissestien 67', City='Sandnes'  WHERE LastName='Tjessem' AND FirstName='Jakob' |

The "Persons" table will now look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Johan | Bakken 2 | Stavanger |
| 5 | Tjessem | Jakob | Nissestien 67 | Sandnes |

## SQL UPDATE Warning

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

|  |
| --- |
| UPDATE Persons  SET Address='Nissestien 67', City='Sandnes' |

The "Persons" table would have looked like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Nissestien 67 | Sandnes |
| 2 | Svendson | Tove | Nissestien 67 | Sandnes |
| 3 | Pettersen | Kari | Nissestien 67 | Sandnes |
| 4 | Nilsen | Johan | Nissestien 67 | Sandnes |
| 5 | Tjessem | Jakob | Nissestien 67 | Sandnes |

# 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 |

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

## SQL DELETE Example

The "Persons" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Johan | Bakken 2 | Stavanger |
| 5 | Tjessem | Jakob | Nissestien 67 | Sandnes |

Now we want to delete the person "Tjessem, Jakob" in the "Persons" table.

We use the following SQL statement:

|  |
| --- |
| DELETE FROM Persons  WHERE LastName='Tjessem' AND FirstName='Jakob' |

The "Persons" table will now look like this:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P\_Id** | **LastName** | **FirstName** | **Address** | **City** |
| 1 | Hansen | Ola | Timoteivn 10 | Sandnes |
| 2 | Svendson | Tove | Borgvn 23 | Sandnes |
| 3 | Pettersen | Kari | Storgt 20 | Stavanger |
| 4 | Nilsen | Johan | Bakken 2 | Stavanger |

## Delete All Rows

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!

## Test your SQL Skills

On this page you can test your SQL skills.

We will use the **Customers** table in the Northwind database:

|  |  |  |  |
| --- | --- | --- | --- |
| **CompanyName** | **ContactName** | **Address** | **City** |
| Alfreds Futterkiste | Maria Anders | Obere Str. 57 | Berlin |
| Berglunds snabbköp | Christina Berglund | Berguvsvägen 8 | Luleå |
| Centro comercial Moctezuma | Francisco Chang | Sierras de Granada 9993 | México D.F. |
| Ernst Handel | Roland Mendel | Kirchgasse 6 | Graz |
| FISSA Fabrica Inter. Salchichas S.A. | Diego Roel | C/ Moralzarzal, 86 | Madrid |
| Galería del gastrónomo | Eduardo Saavedra | Rambla de Cataluña, 23 | Barcelona |
| Island Trading | Helen Bennett | Garden House Crowther Way | Cowes |
| Königlich Essen | Philip Cramer | Maubelstr. 90 | Brandenburg |
| Laughing Bacchus Wine Cellars | Yoshi Tannamuri | 1900 Oak St. | Vancouver |
| Magazzini Alimentari Riuniti | Giovanni Rovelli | Via Ludovico il Moro 22 | Bergamo |
| North/South | Simon Crowther | South House 300 Queensbridge | London |
| Paris spécialités | Marie Bertrand | 265, boulevard Charonne | Paris |
| Rattlesnake Canyon Grocery | Paula Wilson | 2817 Milton Dr. | Albuquerque |
| Simons bistro | Jytte Petersen | Vinbæltet 34 | København |
| The Big Cheese | Liz Nixon | 89 Jefferson Way Suite 2 | Portland |
| Vaffeljernet | Palle Ibsen | Smagsløget 45 | Århus |
| Wolski Zajazd | Zbyszek Piestrzeniewicz | ul. Filtrowa 68 | Warszawa |

To preserve space, the table above is a subset of the Customers table used in the example below.

## Try it Yourself

To see how SQL works, you can copy the SQL statements below and or you can make your own SQL statements.

|  |
| --- |
| SELECT \* FROM customers |

|  |
| --- |
| SELECT CompanyName, ContactName  FROM customers |

|  |
| --- |
| SELECT \* FROM customers  WHERE CompanyName LIKE 'a%' |

|  |
| --- |
| SELECT CompanyName, ContactName  FROM customers  WHERE CompanyName > 'a' |

D:\Different file\fantu\www.w3schools.com\www.w3schools.com\sql\lamp.gifWhen using SQL on text data, "alfred" is greater than "a" (like in a dictionary).

|  |
| --- |
| SELECT CompanyName, ContactName  FROM customers  WHERE CompanyName > 'g'  AND ContactName > 'g' |

SELECT \* FROM customers