A Crash Course in SQL

More available at w3schools.com

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 became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

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/ISO 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.

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.

Look at the "Customers" table:

```
SELECT * FROM Customers;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil
16	Consolidated Holdings	Elizabeth Brown	Berkeley Gardens 12 Brewery	London	WX1 6LT	UK
17	Drachenblut Delikatessend	Sven Ottlieb	Walserweg 21	Aachen	52066	Germany

Fields, rows and columns

- Every table is broken up into smaller entities called **fields**.
- The fields in the Customers table consist of CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country.
- A **field** is a column in a table that is designed to maintain specific information about every record in the table.
- A **record**, also called a **row**, is each individual entry that exists in a table. For example, there are 91 records in the above Customers table.
- A **record** is a horizontal entity in a table.
- A **column** is a vertical entity in a table that contains all information associated with a specific field in a table.

SQL

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

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

- 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.

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 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

SQL **Select** Statement

The SQL SELECT Statement

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

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

```
SELECT column1, column2, … FROM table_name;
```

Here, column1, column2, ... are the field names of the table you want to select data from.

Example:

```
SELECT CustomerName, ContactName FROM Customers;
```

The SQL SELECT Statement

Example:

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

CustomerName	City
Alfreds Futterkiste	Berlin
Ana Trujillo Emparedados y helados	México D.F.
Antonio Moreno Taquería	México D.F.
Around the Horn	London
Berglunds snabbköp	Luleå
Blauer See Delikatessen	Mannheim

SQL Select Statement

If you want to select all the fields available in the table, use the following syntax:

```
SELECT * FROM table_name;
```

Example

```
SELECT * FROM Customers;
```

SQL **Select Distinct** Statement

The SQL SELECT DISTINCT Statement

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

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

```
SELECT DISTINCT column1, column2, ...FROM table_name;
```

Example

```
SELECT DISTINCT column1, column2, ...FROM table_name;
```

A:

City
Berlin
México D.F.
London
Luleå
Mannheim

B:

City
Berlin
México D.F.
México D.F.
London
Luleå
Mannheim

The WHERE Clause

The SQL WHERE Clause

The `WHERE` clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

Note: The `WHERE` clause is not only used in `SELECT` statements, it is also used in `UPDATE`, `DELETE`, etc.!

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';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
58	Pericles Comidas clásicas	Guillermo Fernández	Calle Dr. Jorge Cash 321	México D.F.	05033	Mexico
80	Tortuga Restaurante	Miguel Angel Paolino	Avda. Azteca 123	México D.F.	05033	Mexico

The WHERE Clause

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

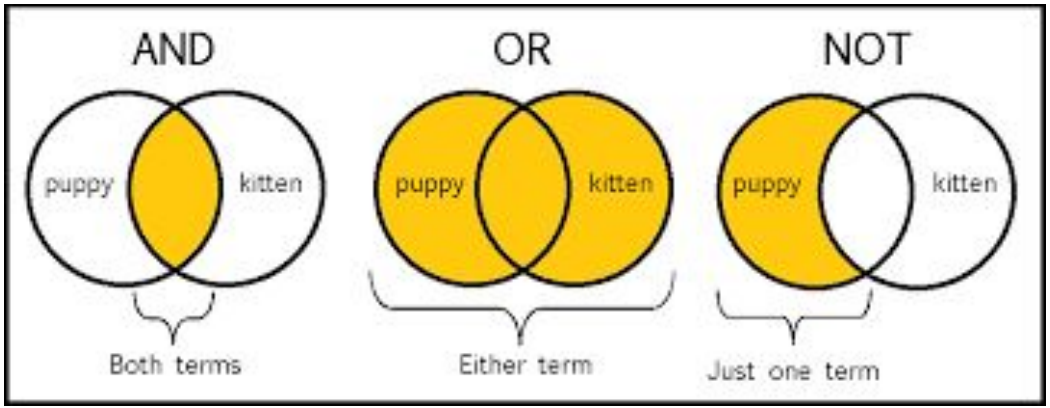
```
SELECT * FROM Customers
WHERE CustomerID=1;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

Operators in The WHERE Clause

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

AND, OR and NOT Operators



The SQL AND, OR and NOT Operators

The `WHERE` clause can be combined with `AND`, `OR`, and `NOT` operators.

The `AND` and `OR` operators are used to filter records based on more than one condition:

- The `AND` operator displays a record if all the conditions separated by `AND` are TRUE.
- The `OR` operator displays a record if any of the conditions separated by `OR` is TRUE.

The `NOT` operator displays a record if the condition(s) is NOT TRUE.

AND Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

AND Example

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

Example

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany

OR Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

OR Example

The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "München":

Example

```
SELECT * FROM Customers

WHERE City='Berlin' OR City='München';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
25	Frankenversand	Peter Franken	Berliner Platz 43	München	80805	Germany

NOT Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

```
SELECT * FROM Customers
WHERE NOT Country='Germany' AND NOT Country='Mexico' and NOT Country='UK' and NOT Country='Sweden'and NOT Country='France'and NOT
Country='Canada'and NOT Country='Brazil'and NOT Country='Spain'and NOT Country='Argentina'and NOT Country='Switzerland'and NOT
Country='USA'and NOT Country='Venezuela'and NOT Country='Italy'and NOT Country='Belgium';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
28	Furia Bacalhau e Frutos do Mar	Lino Rodriguez	Jardim das rosas n. 32	Lisboa	1675	Portugal
37	Hungry Owl All-Night Grocers	Patricia McKenna	8 Johnstown Road	Cork		Ireland
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
60	Princesa Isabel Vinhoss	Isabel de Castro	Estrada da saúde n. 58	Lisboa	1756	Portugal
70	Santé Gourmet	Jonas Bergulfsen	Erling Skakkes gate 78	Stavern	4110	Norway
73	Simons bistro	Jytte Petersen	Vinbæltet 34	København	1734	Denmark
83	Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus	8200	Denmark
87	Wartian Herkku	Pirkko Koskitalo	Torikatu 38	Oulu	90110	Finland
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland

Exercise:

Select all records where the `City` column has the value 'Berlin' and the `PostalCode` column has the value 12209.

```
      * FROM Customers
     City = 'Berlin'
   AND          = 12209;
```

Exercise:

Select all records where the `City` column has the value 'Berlin' and the `PostalCode` column has the value 12209.

```
      * FROM Customers
     City = 'Berlin'
   =  = 12209;
```

```
SELECT * FROM Customers
WHERE City = 'Berlin'
AND PostalCode = 12209;
```

ORDER BY

The SQL ORDER BY Keyword

The `ORDER BY` keyword is used to sort the result-set in ascending or descending order.

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

ORDER BY Syntax

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

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;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium

...

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;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
33	GROSELLA-Restaurante	Manuel Pereira	5ª Ave. Los Palos Grandes	Caracas	1081	Venezuela
35	HILARIÓN-Abastos	Carlos Hernández	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	5022	Venezuela
46	LILA-Supermercado	Carlos González	Carrera 52 con Ave. Bolívar #65-98 Llano Largo	Barquisimeto	3508	Venezuela
47	LINO-Delicateses	Felipe Izquierdo	Ave. 5 de Mayo Porlamar	I. de Margarita	4980	Venezuela

...

ORDER BY Several Columns Example

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" and the "CustomerName" column. This means that it orders by Country, but if some rows have the same Country, it orders them by CustomerName:

Example

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
50	Maison Dewey	Catherine Dewey	Rue Joseph-Bens 532	Bruxelles	B-1180	Belgium

...

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;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
64	Rancho grande	Sergio Gutiérrez	Av. del Libertador 900	Buenos Aires	1010	Argentina
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
59	Piccolo und mehr	Georg Pipps	Geislweg 14	Salzburg	5020	Austria
20	Ernst Handel	Roland Mendel	Kirchgasse 6	Graz	8010	Austria
76	Suprêmes délices	Pascale Cartrain	Boulevard Tirou, 255	Charleroi	B-6000	Belgium

...

INSERT

The SQL INSERT INTO Statement

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

INSERT INTO Syntax

It is possible to write the `INSERT INTO` statement in two ways:

1. Specify both the column names and the values to be inserted:

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

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the `INSERT INTO` syntax would be as follows:

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

INSERT INTO Example

The following SQL statement inserts a new record in the "Customers" table:

Example

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

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 record, but only insert data in the "CustomerName", "City", and "Country" columns (CustomerID will be updated automatically):

Example

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

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	null	null	Stavanger	null	Norway

NULL

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the `IS NULL` and `IS NOT NULL` operators instead.

IS NULL Syntax

```
SELECT column_names

FROM table_name

WHERE column_name IS NULL;
```

IS NOT NULL Syntax

```
SELECT column_names

FROM table_name

WHERE column_name IS NOT NULL;
```

The IS NULL Operator

The `IS NULL` operator is used to test for empty values (NULL values).

The following SQL lists all customers with a NULL value in the "Address" field:

Example

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

Tip: Always use IS NULL to look for NULL values.

The IS NULL Operator

The `IS NULL` operator is used to test for empty values (NULL values).

The following SQL lists all customers with a NULL value in the "Address" field:

Example

```
SELECT CustomerName, ContactName, Address
FROM Customers
WHERE Address IS NULL;
```

CustomerName	ContactName	Address
Cardinal	*null*	*null*

The IS NOT NULL Operator

The `IS NOT NULL` operator is used to test for non-empty values (NOT NULL values).

The following SQL lists all customers with a value in the "Address" field:

Example

```
SELECT CustomerName, ContactName, AddressFROM Customers WHERE Address IS NOT NULL;
```

CustomerName	ContactName	Address
Alfreds Futterkiste	Maria Anders	Obere Str. 57
Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222
Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312
Around the Horn	Thomas Hardy	120 Hanover Sq.
Berglunds snabbköp	Christina Berglund	Berguvsvägen 8

The SQL UPDATE Statement

The `UPDATE` statement is used to modify the existing records in a table.

UPDATE Syntax

```
UPDATE table_name

SET column1 = value1, column2 = value2, ...

WHERE condition;
```

Note: Be careful when updating records in a table! Notice the `WHERE` clause in the `UPDATE` statement. The `WHERE` clause specifies which record(s) that should be updated. If you omit the `WHERE` clause, all records in the table will be updated!

UPDATE

UPDATE Table

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

Example

```
UPDATE Customers SET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

UPDATE Multiple Records

It is the `WHERE` clause that determines how many records will be updated.

The following SQL statement will update the ContactName to "Juan" for all records where country is "Mexico":

Example

```
UPDATE Customers SET ContactName='Juan' WHERE Country='Mexico';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Alfred Schmidt	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Update Warning!

Be careful when updating records. If you omit the `WHERE` clause, ALL records will be updated!

Example

```
UPDATE Customers SET ContactName='Juan';
```

Update Warning!

Be careful when updating records. If you omit the `WHERE` clause, ALL records will be updated!

Example

```
UPDATE Customers SET ContactName='Juan';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Juan	Obere Str. 57	Frankfurt	12209	Germany
2	Ana Trujillo Emparedados y helados	Juan	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Juan	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Juan	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Juan	Berguvsvägen 8	Luleå	S-958 22	Sweden

Exercise:

Update the `City` column of all records in the `Customers` table.

```
        Customers
    City = 'Oslo';
```

Exercise:

Update the `City` column of all records in the `Customers` table.

```
UPDATE Customers
SET City = 'Oslo';
```

DELETE

The SQL DELETE Statement

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

DELETE Syntax

```
DELETE FROM table_name WHERE condition;
```

Note: Be careful when deleting records in a table! Notice the `WHERE` clause in the `DELETE` statement. The `WHERE` clause specifies which record(s) should be deleted. If you omit the `WHERE` clause, all records in the table will be deleted!

SQL DELETE Example

The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

Example

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden

Delete All Records

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;
```

The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

Example

```
DELETE FROM Customers;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country

Exercise:

Delete all the records from the `Customers` table where the `Country` value is 'Norway'.

```
    Customers
WHERE Country = 'Norway';
```

Exercise:

Delete all the records from the `Customers` table where the `Country` value is 'Norway'.

```
          Customers
    Country = 'Norway';
```

Exercise:

Delete all the records from the `Customers` table where the `Country` value is 'Norway'.

```
DELETE FROM Customers
WHERE Country = 'Norway';
```

TOP

The SQL SELECT TOP Clause

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

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

Note: Not all database systems support the `SELECT TOP` clause. MySQL supports the `LIMIT` clause to select a limited number of records, while Oracle uses `FETCH FIRST n ROWS ONLY` and `ROWNUM`.

MySQL Syntax:

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

TOP Example:

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

Example

```
SELECT TOP 3 * FROM Customers;
```

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

Example

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
17	Drachenblut Delikatessend	Sven Ottlieb	Walserweg 21	Aachen	52066	Germany

MIN and MAX functions

The SQL MIN() and MAX() Functions

The `MIN()` function returns the smallest value of the selected column.

The `MAX()` function returns the largest value of the selected column.

MIN() Syntax

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

MAX() Syntax

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

MIN() Example

The following SQL statement finds the price of the cheapest product:

Example

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

SmallestPrice
2.5

MAX() Example

The following SQL statement finds the price of the most expensive product:

Example

```
SELECT MAX(Price) AS LargestPrice
FROM Products;
```

LargestPrice
263.5

COUNT(), AVG() and SUM() Functions

The `COUNT()` function returns the number of rows that matches a specified criterion.

COUNT() Syntax

```
SELECT COUNT(column_name)

FROM table_name

WHERE condition;
```

Count

The SQL COUNT keyword allows us to do exactly that → count the number of results returned

Example:

```
SELECT COUNT(DISTINCT Country) FROM Customers;
```

COUNT(DISTINCT Country)
5

The `AVG()` function returns the average value of a numeric column.

AVG() Syntax

```
SELECT AVG(column_name)

FROM table_name

WHERE condition;
```

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	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Example

```
SELECT AVG(Price) FROM Products;
```

AVG(Price)
28.866363636363637

The `SUM()` function returns the total sum of a numeric column.

SUM() Syntax

```
SELECT SUM(column_name)

FROM table_name

WHERE condition;
```

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	2	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	2	2	36 boxes	21.35

Example

```
SELECT SUM(Quantity) FROM OrderDetails;
```

SUM(Quantity)
12743

Exercise:

Use the correct function to return the number of records that have the `Price` value set to 18.

```
SELECT      (*)
FROM Products
      Price = 18;
```

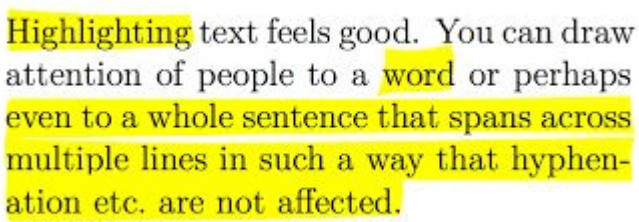
Exercise:

Use the correct function to return the number of records that have the `Price` value set to 18.

```
SELECT      (*)
FROM Products
      Price = 18;
```

```
SELECT COUNT(*)
FROM Products
WHERE Price = 18;
```

LIKE Operator



The SQL LIKE Operator

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

There are two wildcards often used in conjunction with the `LIKE` operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

The percent sign and the underscore can also be used in combinations!

LIKE Syntax

```
SELECT column1, column2, ...

FROM table_name

WHERE columnN LIKE pattern;
```

Tip: You can also combine any number of conditions using `AND` or `OR` operators.

Here are some examples showing different `LIKE` operators with '%' and '_' wildcards:

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

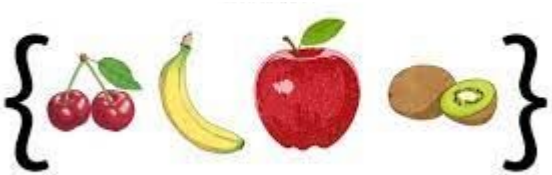
The following SQL statement selects all customers with a CustomerName starting with "a":

Example

```
SELECT * FROM Products
```

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
39	Chartreuse verte	18	1	750 cc per bottle	18

IN Operator



The SQL IN Operator

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

The `IN` operator is a shorthand for multiple `OR` conditions.

IN Syntax

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

or:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

IN Example:

```
SELECT * FROM Products WHERE Price IN (10, 15);
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10
70	Outback Lager	7	1	24 - 355 ml bottles	15
73	Röd Kaviar	17	8	24 - 150 g jars	15
74	Longlife Tofu	4	7	5 kg pkg.	10

IN Example:

```
SELECT * FROM Products WHERE Price IN (10, 15);
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10
70	Outback Lager	7	1	24 - 355 ml bottles	15
73	Röd Kaviar	17	8	24 - 150 g jars	15
74	Longlife Tofu	4	7	5 kg pkg.	10

Another way to say this in SQL?

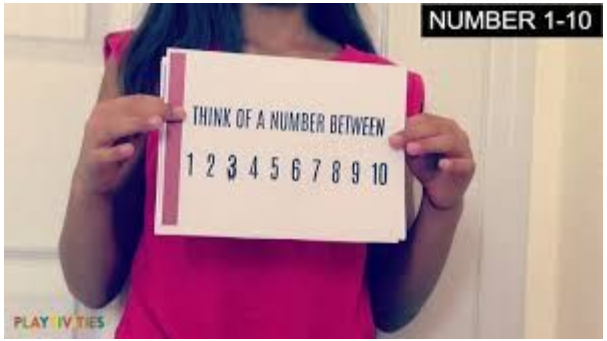
IN Example:

```
SELECT * FROM Products WHERE Price IN (10, 15);
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10
70	Outback Lager	7	1	24 - 355 ml bottles	15
73	Röd Kaviar	17	8	24 - 150 g jars	15
74	Longlife Tofu	4	7	5 kg pkg.	10

```
SELECT * FROM Products WHERE Price=10 OR Price=15;
```

BETWEEN Operator



The SQL BETWEEN Operator

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

The `BETWEEN` operator is inclusive: begin and end values are included.

BETWEEN Syntax

```
SELECT column_name(s)

FROM table_name

WHERE column_name BETWEEN value1 AND value2;
```

Example

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 12;
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
21	Sir Rodney's Scones	8	3	24 pkgs. x 4 pieces	10
46	Spegesild	21	8	4 - 450 g glasses	12
74	Longlife Tofu	4	7	5 kg pkg.	10

BETWEEN Text Values Example

The following SQL statement selects all products with a ProductName between Carnarvon Tigers and Mozzarella di Giovanni:

Example

```
SELECT * FROM Products

WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Chartreuse%'

ORDER BY ProductName;
```

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
18	Carnarvon Tigers	7	8	16 kg pkg.	62.5
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
39	Chartreuse verte	18	1	750 cc per bottle	18

We’ve covered:

- SELECT
- WHERE
- AND, OR, NOT
- Order By
- Insert Into
- Null Values
- Delete
- Select Top
- Min and Max
- Count, Avg, Sum
- Like
- In
- Between

Next Time:

- Creating Tables
- SQL Constraints
- Primary Keys
- Foreign Keys