﻿The way the SQL works is you have these keywords such as SELECT or FROM or CREATE or TABLE and you tend to write them in uppercase and using this Structured Query Language, we can create tables, manipulate them, update, destroy, etc.

Now with every single type of database the main things that you'll be doing with it is simply create, read, update, destroy. Or in database lingo it's known as CRUD.

So, for every single database the first thing to do is to get yourself used to doing CRUD using that particular database.

SQL Create Table. And here is the syntax for how you would create a table using SQL.

So, the key word here are in all caps and that's CREATE TABL and then we provide a name for the table and then we open a set of parentheses and inside the parentheses, we detail the names of each column and the data type that it will contain and these are separated by commas.

For ex, the first one is going to be an ID and this is going to be an integer data type and that is going to store the unique id or the primary key for our products table. So, we'll be able to identify each row by its id. The next one we're going to call name and this is going to be a string.

Now remember we're not writing JavaScript here anymore so there's no colons and there's actually just a space between the name and the data type and there's a comma in between each one of these columns. The most commonly used ones are things such as string or text or characters of a particular size. So, if you said cha (255) then it can only store up to 255 characters and you can limit your data in that way

The final thing that we're going to add to our schema or the structure of our table is something called a primary key. Again, scrolling through the documentation, you will find a section on SQL primary keys. And what this does is it allows a particular column to uniquely identify each record in a database. So that means that this record here with name of pen and price of 1.20 will be uniquely identified by this 1 and there won't be another product with the id of 1.

So, whenever we say products with id of 1 then it refers to one specific record. And to do this we have to set a particular column as the primary key for the table. In order to do that using SQL you can see that we have to write the word primary key, so these are special keywords, and then inside a set of parentheses we specify the field that is going to be the primary key which in our case is going to be this one called id. So, let's go ahead and add primary key and a set of parentheses and then the name of the field on the column that is going to be the primary key.

So, in our case it's again id. Now if you see in this documentation something else you might notice is that for when they created their id field, they added another keyword next to it which is not null. This guarantees that whenever new records are being created inside this table if the record doesn't provide an ID then it will not allow it to be created so it cannot be null.

## **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, ...);

SQL Select

This keyword allows to read data from tables.

Format for SQL Select Statement for columns is something like

SELECT [field1] , [field2] FROM ‘[tablename]’;

Format for selecting rows

SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

Example

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

## **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;

Example

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

## **SQL ALTER TABLE Statement**

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

## **ALTER TABLE - ADD Column**

To add a column in a table, use the following syntax:

ALTER TABLE table\_name  
ADD column\_name datatype;

The following SQL adds an "Email" column to the "Customers" table:

### Example

ALTER TABLE Customers  
ADD Email varchar(255);

## **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';

## **SQL FOREIGN KEY Constraint**

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/sql/sql_primarykey.asp) in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

Look at the following two tables:

### Persons Table

|  |  |  |  |
| --- | --- | --- | --- |
| **PersonID** | **LastName** | **FirstName** | **Age** |
| 1 | Hansen | Ola | 30 |
| 2 | Svendson | Tove | 23 |
| 3 | Pettersen | Kari | 20 |

### Orders Table

|  |  |  |
| --- | --- | --- |
| **OrderID** | **OrderNumber** | **PersonID** |
| 1 | 77895 | 3 |
| 2 | 44678 | 3 |
| 3 | 22456 | 2 |
| 4 | 24562 | 1 |

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

The "PersonID" column in the "Persons" table is the PRIMARY KEY in the "Persons" table.

The "PersonID" column in the "Orders" table is a FOREIGN KEY in the "Orders" table.

The FOREIGN KEY constraint prevents invalid data from being inserted into the foreign key column, because it has to be one of the values contained in the parent table.

## **SQL FOREIGN KEY on CREATE TABLE**

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

**MySQL:**

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

## **SQL INNER JOIN Keyword**

The INNER JOIN keyword selects records that have matching values in both tables.

### INNER JOIN Syntax

SELECT column\_name(s)  
FROM table1  
INNER JOIN table2ON table1.column\_name = table2.column\_name;



## **Demo Database**

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

Below is 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 |

And a selection from the "Customers" table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |

## **SQL INNER JOIN Example**

The following SQL statement selects all orders with customer information:

### Example

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