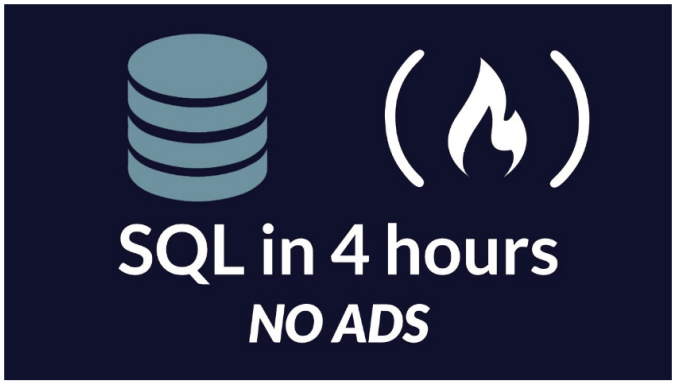
The Best SQL Database Tutorials

SQL stands for Structured Query Language. It is the most common tool used to manipulate and manage data in a relational database (often referred to as a "SQL database").

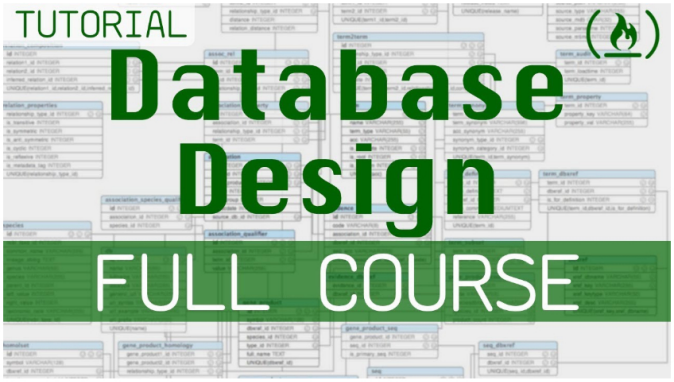
SQL is commonly pronounced "sequel." Its most popular variants are MySQL, PostgreSQL, and SQLite - a version of SQL which is commonly used for prototyping. It introduced the concept of accessing many records with one single command, using SQL Queries.

We recommend starting with freeCodeCamp's 4 hour SQL database tutorial.



We also recommend Harvard CS50's course on databases and SQL.

And if you're feeling up for it, here's an entire 9-hour tutorial on relational database design so you can build your own RDBMS system using SQL.



Some common SQL statements and queries

The SQL Select Statement

Select and From clauses

The SELECT part of a query is normally to determine which columns of the data to show in the results. There are also options you can apply to show data that is not a table column.

This example shows three columns selected from the “student” table and one calculated column. The database stores the studentID, FirstName, and LastName of the student. We can combine the First and the Last name columns to create the FullName calculated column.

```
select studentID, FirstName, LastName, FirstName + ' ' + LastName as FullName
from student;
```

```
+-----------+-------------------+------------+------------------------+
| studentID | FirstName         | LastName   | FullName               |
+-----------+-------------------+------------+------------------------+
|         1 | Monique           | Davis      | Monique Davis          |
|         2 | Teri              | Gutierrez  | Teri Gutierrez         |
|         3 | Spencer           | Pautier    | Spencer Pautier        |
|         4 | Louis             | Ramsey     | Louis Ramsey           |
|         5 | Alvin             | Greene     | Alvin Greene           |
|         6 | Sophie            | Freeman    | Sophie Freeman         |
|         7 | Edgar Frank "Ted" | Codd       | Edgar Frank "Ted" Codd |
|         8 | Donald D.         | Chamberlin | Donald D. Chamberlin   |
|         9 | Raymond F.        | Boyce      | Raymond F. Boyce       |
+-----------+-------------------+------------+------------------------+
9 rows in set (0.00 sec)
```

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a single column it allows only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in

the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the “Age” column when the “Persons” table is created. The CHECK constraint ensures that you can not have any person below 18 years:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255),
    CONSTRAINT CHK_Person CHECK (Age>=18 AND City='Sandnes')
);
```

SQL CHECK on ALTER TABLE

To create a CHECK constraint on the “Age” column when the table is already created, use the following SQL:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CHECK (Age>=18);
```

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

MySQL / SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
ADD CONSTRAINT CHK_PersonAge CHECK (Age>=18 AND City='Sandnes');
```

DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

SQL Server / Oracle / MS Access:

```
ALTER TABLE Persons
DROP CONSTRAINT CHK_PersonAge;
```

MySQL:

```
ALTER TABLE Persons
DROP CHECK CHK_PersonAge;
```

SQL Where Clause

`WHERE` Clause (and/or, `IN`, `BETWEEN`, and `LIKE`)

The `WHERE` clause is used to limit the number of rows returned.

In this case all five of these will be used is a somewhat ridiculous `WHERE` clause.

Here is the current full student list to compare to the `WHERE` clause result set:

```
select studentID, FullName, sat_score, rcd_updated from student;
```

```
+-----------+-----------------------+-----------+---------------------+
| studentID | FullName              | sat_score | rcd_updated         |
+-----------+-----------------------+-----------+---------------------+
|         1 | Monique Davis         |       400 | 2017-08-16 15:34:50 |
|         2 | Teri Gutierrez        |       800 | 2017-08-16 15:34:50 |
|         3 | Spencer Pautier       |      1000 | 2017-08-16 15:34:50 |
|         4 | Louis Ramsey          |      1200 | 2017-08-16 15:34:50 |
|         5 | Alvin Greene          |      1200 | 2017-08-16 15:34:50 |
|         6 | Sophie Freeman        |      1200 | 2017-08-16 15:34:50 |
|         7 | Edgar Frank "Ted" Codd |      2400 | 2017-08-16 15:35:33 |
|         8 | Donald D. Chamberlin  |      2400 | 2017-08-16 15:35:33 |
|         9 | Raymond F. Boyce      |      2400 | 2017-08-16 15:35:33 |
+-----------+-----------------------+-----------+---------------------+
9 rows in set (0.00 sec)
```

Rows will be presented that...

- `WHERE` Student IDs are between 1 and 5 (inclusive)
- `OR` studentID = 8

Here's an updated query, where any record that has an SAT

score that's in this list (1000, 1400) will not be presented:

```
select  studentID, FullName, sat_score, recordUpdated
from    student
where   (studentID between 1 and 5 or studentID = 8)
        and
        sat_score NOT in (1000, 1400);
```

```
+-----------+---------------------+-----------+---------------------+
| studentID | FullName            | sat_score | rcd_updated         |
+-----------+---------------------+-----------+---------------------+
|         1 | Monique Davis       |       400 | 2017-08-16 15:34:50 |
|         2 | Teri Gutierrez      |       800 | 2017-08-16 15:34:50 |
|         4 | Louis Ramsey        |      1200 | 2017-08-16 15:34:50 |
|         5 | Alvin Greene        |      1200 | 2017-08-16 15:34:50 |
|         8 | Donald D. Chamberlin |      2400 | 2017-08-16 15:35:33 |
+-----------+---------------------+-----------+---------------------+
5 rows in set (0.00 sec)
```

*As with all of these SQL things there is MUCH MORE to them than what's in this introductory guide.

I hope this at least gives you enough to get started.

Please see the manual for your database manager and have fun trying different options yourself.

SQL Update Statement

To update a record in a table you use the `UPDATE` statement.

Be careful. You can update all records of the table or just a few. Use the `WHERE` condition to specify which records you want to update. It is possible to update one or more columns at a time. The syntax is:

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

Here is an example updating the Name of the record with Id 4:

```
UPDATE Person
SET Name = "Elton John"
WHERE Id = 4;
```

You can also update columns in a table by using values from other tables. Use `JOIN` clause to get data from multiple tables. The syntax is:

```
UPDATE table_name1
SET table_name1.column1 = table_name2.columnA
    table_name1.column2 = table_name2.columnB
FROM table_name1
JOIN table_name2 ON table_name1.ForeignKey = table_name2.Key
```

Here is an example updating Manager of all records:

```
UPDATE Person
SET Person.Manager = Department.Manager
FROM Person
JOIN Department ON Person.DepartmentID = Department.ID
```

What an Update query can do

An update query gives the DBA or SQL-using programmer the ability to update many records with one command.

Important Safety Tip! Always have a backup copy of what you are about to change BEFORE you change it!

This section will:

- add a new field to the student table
- test the logic to update that field with a school assigned email address
- update the new field.

Here is the student table as we start this process:

```
SELECT * FROM student;
```

```
+-----------+------------------------+-----------+------------------+----------------
| studentID | FullName               | sat_score | programOfStudy   | rcd_Created
+-----------+------------------------+-----------+------------------+----------------
|         1 | Monique Davis          |       400 | Literature       | 2017-08-16 15:3
|         2 | Teri Gutierrez         |       800 | Programming      | 2017-08-16 15:3
|         3 | Spencer Pautier        |      1000 | Programming      | 2017-08-16 15:3
|         4 | Louis Ramsey           |      1200 | Programming      | 2017-08-16 15:3
|         5 | Alvin Greene           |      1200 | Programming      | 2017-08-16 15:3
|         6 | Sophie Freeman         |      1200 | Programming      | 2017-08-16 15:3
|         7 | Edgar Frank "Ted" Codd |      2400 | Computer Science | 2017-08-16 15:3
|         8 | Donald D. Chamberlin   |      2400 | Computer Science | 2017-08-16 15:3
|         9 | Raymond F. Boyce       |      2400 | Computer Science | 2017-08-16 15:3
+-----------+------------------------+-----------+------------------+----------------
9 rows in set (0.00 sec)
```

Alter the table and add a new field

```
    ALTER TABLE `fcc_sql_guides_database`.`student`
    ADD COLUMN `schoolEmailAdr` VARCHAR(125) NULL AFTER `programOfStudy`;
```

The student table after the alter is executed.

```
mysql> SELECT FullName, sat_score, programOfStudy, schoolEmailAdr FROM student;
+------------------------+-----------+------------------+----------------+
| FullName               | sat_score | programOfStudy   | schoolEmailAdr |
+------------------------+-----------+------------------+----------------+
| Monique Davis          |       400 | Literature       | NULL           |
| Teri Gutierrez         |       800 | Programming      | NULL           |
| Spencer Pautier        |      1000 | Programming      | NULL           |
| Louis Ramsey           |      1200 | Programming      | NULL           |
| Alvin Greene           |      1200 | Programming      | NULL           |
```

```
| Sophie Freeman          |      1200 | Programming      | NULL           |
| Edgar Frank "Ted" Codd |      2400 | Computer Science | NULL           |
| Donald D. Chamberlin   |      2400 | Computer Science | NULL           |
| Raymond F. Boyce       |      2400 | Computer Science | NULL           |
+------------------------+-----------+------------------+----------------+
9 rows in set (0.00 sec)
```

TESTING the logic (VERY important step!)

```
SELECT FullName, instr(FullName," ") AS firstSpacePosition,
concat(substring(FullName,1,instr(FullName," ")-1),"@someSchool.edu") AS schoolEmail
FROM student;
```

```
+------------------------+--------------------+------------------------+
| FullName               | firstSpacePosition | schoolEmail            |
+------------------------+--------------------+------------------------+
| Monique Davis          |                  8 | Monique@someSchool.edu |
| Teri Gutierrez         |                  5 | Teri@someSchool.edu    |
| Spencer Pautier        |                  8 | Spencer@someSchool.edu |
| Louis Ramsey           |                  6 | Louis@someSchool.edu   |
| Alvin Greene           |                  6 | Alvin@someSchool.edu   |
| Sophie Freeman         |                  7 | Sophie@someSchool.edu  |
| Edgar Frank "Ted" Codd |                  6 | Edgar@someSchool.edu   |
| Donald D. Chamberlin   |                  7 | Donald@someSchool.edu  |
| Raymond F. Boyce       |                  8 | Raymond@someSchool.edu |
+------------------------+--------------------+------------------------+
9 rows in set (0.00 sec)
```

A note about concat(): in MySQL this command is used to combined strings, not so in other SQL versions (check your manual). In this usage it works like this: The substring of the FullName field up to but not including the first space is combined with "@someSchool.edu". In the real world this would HAVE TO be much more complex and you would need to ensure that the email address is unique.

Doing the update

We'll pretend that this is what we want and update the table with this information:

```
UPDATE student SET schoolEmailAdr = concat(substring(FullName,1,instr(FullName," ")-1
WHERE schoolEmailAdr is NULL;
```

Success!

```
mysql> SELECT FullName, sat_score, programOfStudy, schoolEmailAdr FROM student;
+------------------------+-----------+------------------+------------------------+
| FullName               | sat_score | programOfStudy   | schoolEmailAdr         |
+------------------------+-----------+------------------+------------------------+
| Monique Davis          |       400 | Literature       | Monique@someSchool.edu |
| Teri Gutierrez         |       800 | Programming      | Teri@someSchool.edu    |
| Spencer Pautier        |      1000 | Programming      | Spencer@someSchool.edu |
| Louis Ramsey           |      1200 | Programming      | Louis@someSchool.edu   |
| Alvin Greene           |      1200 | Programming      | Alvin@someSchool.edu   |
| Sophie Freeman         |      1200 | Programming      | Sophie@someSchool.edu  |
| Edgar Frank "Ted" Codd |      2400 | Computer Science | Edgar@someSchool.edu   |
| Donald D. Chamberlin   |      2400 | Computer Science | Donald@someSchool.edu  |
| Raymond F. Boyce       |      2400 | Computer Science | Raymond@someSchool.edu |
+------------------------+-----------+------------------+------------------------+
9 rows in set (0.00 sec)
```

)

If this article was helpful, tweet it.

Learn to code for free. freeCodeCamp's open source curriculum has helped more than 40,000 people get jobs as developers.

Get started