

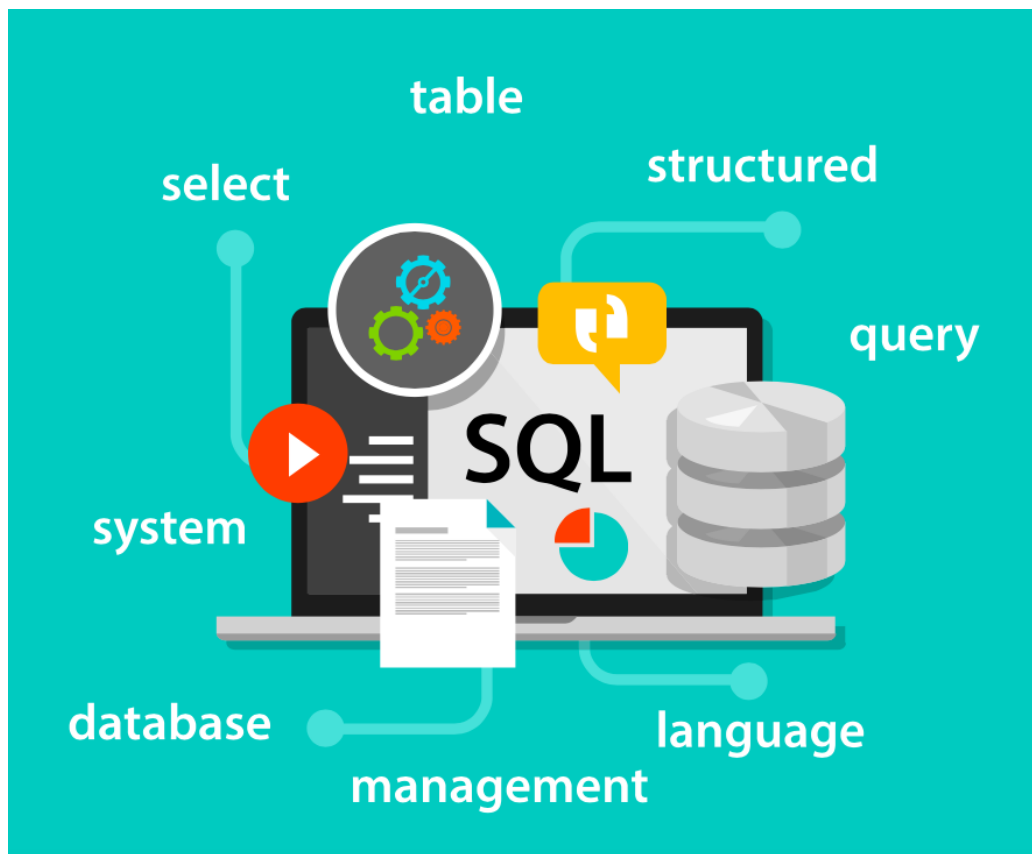
What is SQL?



**SQL**

## Background of MySQL

SQL was developed by IBM researchers in the 1970s. SQL (Structured Query Language) is a programming language we use to communicate to databases. Examples are find all users, add a new user and delete a user. A database is a collection of data. It is a system of accessing and changing the data. It is also a computerized data with an accessible interface. MySQL, SQLite, PostgreSQL, Oracle etc. are all relational databases that use SQL. It is easy to switch to another database that also uses SQL. SQL is easily accessible on many platforms and easy to use. You can install SQL on a PC, on a MAC, and on Cloud 9. SQL is used everywhere especially in businesses.



## Basic Commands in SQL

There are basic commands in SQL that you should know. The basic commands to create a database and a table. A database is a group of tables. Tables hold data.

- To create a database the syntax is:

`create database <database name>;`

For example:

```
mysql> CREATE DATABASE DogApp;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE DATABASE soap_soft;
Query OK, 1 row affected (0.00 sec)

mysql> CREATE DATABASE hello_world;
Query OK, 1 row affected (0.00 sec)
```

- To show what databases are in the system the syntax is:

`show databases;`

For example:

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| Course_Registration |
| DogApp |
| MyApp |
| hello_world |
| metastore |
| movies |
| mysql |
| performance_schema |
| soap_soft |
| sys |
+-----+
11 rows in set (0.00 sec)
```

- To select a database to work with the syntax is:

```
select database ();
```

For example:



```
mysql> SELECT database();
+-----+
| database() |
+-----+
| DogApp     |
+-----+
1 row in set (0.00 sec)
```

- To use the selected database and create a table for it the syntax is:

```
use <database name>;
```

For example:



```
mysql> USE DogApp;
Database changed
```

- To drop a database the syntax is:

```
drop database <database name>;
```

For example:

```
mysql> DROP DATABASE hello_world;
Query OK, 0 rows affected (0.00 sec)

mysql> DROP DATABASE DogApp;
Query OK, 0 rows affected (0.00 sec)

mysql> DROP DATABASE soap_soft;
Query OK, 0 rows affected (0.00 sec)
```

- To create a table for a database the syntax is:

create table <table name>;

For example:

```
mysql> CREATE TABLE cat (cat_id INT NOT NULL, first_name VARCHAR(100) NOT NULL, breed VARCHAR(100) NOT NULL, age INT NOT NULL, PRIMARY KEY(cat_id));
Query OK, 0 rows affected (0.01 sec)
```

- To show tables in the database the system the syntax is:

show tables;

For example:

```
mysql> show tables;
Empty set (0.00 sec)
```

- To show characterizes of a table you created the syntax is:

describe <table name>;

For example:

```
mysql> DESC cat;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| cat_id     | int(11)       | NO   | PRI | NULL    |      |
| first_name | varchar(100)  | NO   |     | NULL    |      |
| breed      | varchar(100)  | NO   |     | NULL    |      |
| age        | int(11)       | NO   |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

- To drop a table in a database the syntax is:

`drop table <table name>;`

For example:

```
mysql> DROP TABLE cat;
Query OK, 0 rows affected (0.01 sec)
```

- You insert data in the table by using the syntax:

`insert into <tablename> value (<data information>;`

For example:

```

mysql> INSERT INTO cat VALUE ('1', 'Ringo', 'Tabby', '4');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO cat VALUE ('2', 'Cindy', 'Maine Coon', '11');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO cat VALUE ('3', 'Dumbledore', 'Maine Coon', '10');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO cat VALUE ('4', 'Egg', 'Persian', '4');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO cat VALUE ('5', 'Misty', 'Tabby', '13');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO cat VALUE ('6', 'George Micheal', 'Ragdoll', '9');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO cat VALUE ('7', 'Jackson', 'Sphinyx', '7');
Query OK, 1 row affected (0.00 sec)

```

- To show information from table you created you use the syntax:

`select * from <tablename>;`

For example:

```

mysql> SELECT * FROM cat;
+-----+-----+-----+-----+
| cat_id | first_name | breed | age |
+-----+-----+-----+-----+
| 1 | Ringo | Tabby | 4 |
| 2 | Cindy | Maine Coon | 11 |
| 3 | Dumbledore | Maine Coon | 10 |
| 4 | Egg | Persian | 4 |
| 5 | Misty | Tabby | 13 |
| 6 | George Micheal | Ragdoll | 9 |
| 7 | Jackson | Sphinyx | 7 |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

## Basic String Functions in SQL

String Functions are a way of changing a string as an output. There are basic string functions in SQL you should know.

- **concat**

concat combines data for better output. The syntax is:

`select concat (<column>, <another column>) from <tablename>;`

For example:

```
mysql> SELECT
-> CONCAT(author_fname,author_lname)
-> FROM books;
+-----+
| CONCAT(author_fname,author_lname) |
+-----+
| DoveEggers                        |
| NeilGalman                       |
| DonDellilo                       |
+-----+
```

- **substring**

substring works with parts of strings. The syntax is:

`select substring (<column>, <another column>) from <tablename>;`

For example:



```
mysql> SELECT SUBSTRING("Hello World",-7);
+-----+
| SUBSTRING("Hello World",-7) |
+-----+
| o World                      |
+-----+
1 row in set (0.00 sec)

mysql> SELECT title FROM books;
+-----+
| title                                     |
+-----+
| A Heartbreaking Work of Staggering Genius |
| Caroline                               |
| White Noise                             |
+-----+
3 rows in set (0.00 sec)
```

- **replace**

replace replaces parts of strings. The syntax is:

replace (<string>, <old string>, < new string>);

For example:

```
mysql> SELECT REPLACE('Hello World', 'Hell', '%$#@');
+-----+
| REPLACE('Hello World', 'Hell', '%$#@') |
+-----+
| %$#@o World                             |
+-----+
1 row in set (0.00 sec)

mysql> SELECT REPLACE('Hello World', 'l', '7');
+-----+
| REPLACE('Hello World', 'l', '7') |
+-----+
| He77o Wor7d                       |
+-----+
1 row in set (0.00 sec)

mysql> SELECT REPLACE('Hello World', 'o', '0');
+-----+
| REPLACE('Hello World', 'o', '0') |
+-----+
| Hell0 W0rld                       |
+-----+
1 row in set (0.00 sec)

mysql> SELECT REPLACE('Hello World', 'o', '*');
+-----+
| REPLACE('Hello World', 'o', '*') |
+-----+
| Hello W*rld                       |
+-----+
1 row in set (0.00 sec)
```

- **reverse**

reverse reverses parts of strings. The syntax is:

select reverse (<string>);

For example:

```
mysql> SELECT REVERSE('Hello World');
+-----+
| REVERSE('Hello World') |
+-----+
| dlroW olleH           |
+-----+
1 row in set (0.00 sec)

mysql> SELECT REVERSE('meow meow');
+-----+
| REVERSE('meow meow') |
+-----+
| woem woem            |
+-----+
1 row in set (0.00 sec)
```

- **upper and lower**

upper and lower changes a string's case. The syntax is:

select upper or lower (<expression>) from <tablename>;

For example:

```
mysql> SELECT UPPER('Hello World');
+-----+
| UPPER('Hello World') |
+-----+
| HELLO WORLD          |
+-----+
1 row in set (0.00 sec)

mysql> SELECT LOWER('Hello World');
+-----+
| LOWER('Hello World') |
+-----+
| hello world          |
+-----+
1 row in set (0.00 sec)
```

- **distinct**

distinct returns unique values. The syntax is:

select distinct <column name> from <tablename>;

For example:

```
mysql> SELECT DISTINCT author_lname FROM books;
+-----+
| author_lname |
+-----+
| Eggers       |
| Galman       |
| DeLillo      |
+-----+
3 rows in set (0.00 sec)
```

- **order by**

order by orders results. The syntax is:

select <column name> from <tablename>

order by <column name>;

For example:

```
mysql> SELECT released_year FROM books ORDER BY released_year;
+-----+
| released_year |
+-----+
|          1985 |
|          2001 |
|          2003 |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT released_year FROM books ORDER BY released_year DESC;
+-----+
| released_year |
+-----+
|          2003 |
|          2001 |
|          1985 |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT released_year FROM books ORDER BY released_year ASC;
+-----+
| released_year |
+-----+
|          1985 |
|          2001 |
|          2003 |
+-----+
3 rows in set (0.00 sec)
```

- **limit**

limit limits results. The syntax is:

select <column name> from <tablename>

limit <expression>;

For example:

```
mysql> SELECT title FROM books LIMIT 1;
+-----+
| title |
+-----+
| A Heartbreaking Work of Staggering Genius |
+-----+
1 row in set (0.00 sec)

mysql> SELECT title FROM books LIMIT 3;
+-----+
| title |
+-----+
| A Heartbreaking Work of Staggering Genius |
| Caroline |
| White Noise |
+-----+
3 rows in set (0.00 sec)
```

- like

like helps you with better searching The syntax is:

select <column name> from <tablename>

where <column name> like <pattern>;

For example:

```
mysql> SELECT title FROM books WHERE title LIKE 'the';  
Empty set (0.00 sec)  
  
mysql> SELECT title FROM books WHERE title LIKE '%the';  
Empty set (0.00 sec)  
  
mysql> SELECT title FROM books WHERE title LIKE '%the%';  
Empty set (0.00 sec)  
  
mysql> SELECT title, author_fname FROM books WHERE author_fname LIKE '%da%';  
Empty set (0.00 sec)  
  
mysql> SELECT title, author_fname FROM books WHERE author_fname LIKE 'da%';  
Empty set (0.00 sec)
```

## Count Functions in SQL

- **count**

count returns the number of rows that match the specific statement or number of occurrences of a substring. The syntax is:

select count (<column name>) from <table name>

where <condition>;

For example:

```
mysql> SELECT COUNT(DISTINCT author_fname) FROM books;
+-----+
| COUNT(DISTINCT author_fname) |
+-----+
| 3 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(DISTINCT author_lname) FROM books;
+-----+
| COUNT(DISTINCT author_lname) |
+-----+
| 3 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT COUNT(DISTINCT author_lname, author_fname) FROM books;
+-----+
| COUNT(DISTINCT author_lname, author_fname) |
+-----+
| 3 |
+-----+
1 row in set (0.00 sec)
```

- **min/max return**

min/max returns the lowest and highest values of a column. The syntax is:

select min or max (<column name>) from <table name>

where <condition>;

For example:

```
mysql> SELECT MIN(pages) FROM books;
+-----+
| MIN(pages) |
+-----+
|         208 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT MAX(pages)
-> FROM books;
+-----+
| MAX(pages) |
+-----+
|         437 |
+-----+
1 row in set (0.00 sec)
```

- **sum**

sum calculates the sum of distinct values. The syntax is:

select sum (<column name>) from <table name>

where <condition>;

For example:

```
mysql> SELECT SUM(released_year) FROM books;
+-----+
| SUM(released_year) |
+-----+
|             5989 |
+-----+
1 row in set (0.00 sec)
```



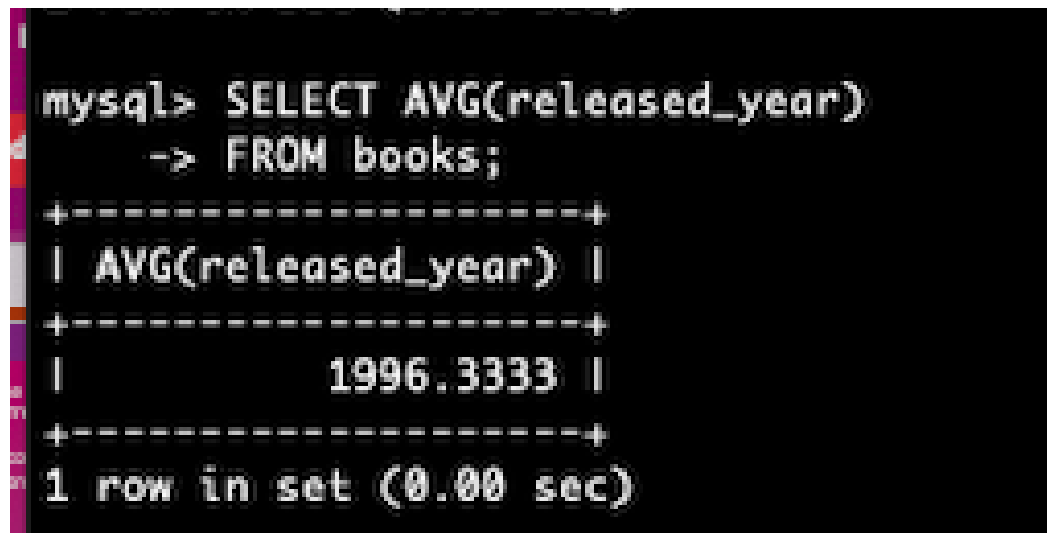
- **avg**

avg find the average of a field in a various record. The syntax is:

select avg (<column name>) from <table name>

where <condition>;

For example:



```
mysql> SELECT AVG(released_year)
-> FROM books;
+-----+
| AVG(released_year) |
+-----+
|          1996.3333 |
+-----+
1 row in set (0.00 sec)
```

- **group by**

group by groups rows with the same values. The syntax is:

select (<column name>) from <table name>

where <condition> group by (<column name>;

For example:

```
/sql> SELECT author_lname, COUNT(*)  
-> FROM books GROUP BY author_lname;
```

```
-----+-----+  
author_lname | COUNT(*) |
```

```
-----+-----+  
Dellilo      |          1 |
```

```
Eggers       |          1 |
```

```
Galman       |          1 |
```

```
-----+-----+  
rows in set (0.00 sec)
```

## Logical Operators in SQL

Logical Operators in SQL are used to check if a statement is true or false.

- **equal/ not equal**

equal/ not equal checks if two expressions are equal or not. The syntax is:

`select * from <table name>`

`where< column name> = or != <expression>;`

For example:

```
mysql> SELECT title, author_lname FROM books WHERE author_lname = 'Harris';
Empty set (0.00 sec)

mysql> SELECT title, author_lname FROM books WHERE author_lname != 'Harris';
+-----+-----+
| title                                | author_lname |
+-----+-----+
| A Heartbreaking Work of Staggering Genius | Eggers      |
| Caroline                             | Galman      |
| White Noise                           | DeLillo     |
+-----+-----+
3 rows in set (0.00 sec)
```

- **like/not**

like/not like checks if two expressions are like each other or not. The syntax is:

`select * from <table name>`

`where< column name> like or not like <expression>;`

- **greater than/ less than**

greater than/ less than checks if two expressions are greater than or less than each other. The syntax is:

select \* from <table name>

where <column name> < or > <expression>;

For example:

```
mysql> SELECT title, released_year FROM books
-> WHERE released_year < 2000;
+-----+-----+
| title      | released_year |
+-----+-----+
| White Noise | 1985          |
+-----+-----+
1 row in set (0.00 sec)
```

- **and**

and returns if all the conditions are met. The syntax is:

select <column name> from <table name>

where <condition 1> and <condition 2>;

- **or**

or returns if any of the conditions are true. The syntax is:

select <column name> from <table name>

where <condition 1> or <condition 2>;

- **between**

between returns values that in the range of each other. The syntax is:

```
select <column name> from <table name>
```

```
where <column name> between <condition 1> and <condition 2>;
```

- **in/not in**

in/not in returns values that are present in the arguments or not. The syntax is:

```
select <column name> from <table name>
```

```
where <column name> in or not in <value>;
```

For example:

```
mysql> SELECT title, released_year FROM books
-> WHERE released_year IN (2017, 1985);
+-----+-----+
| title          | released_year |
+-----+-----+
| White Noise   | 1985          |
+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT title, released_year FROM books
-> WHERE released_year NOT IN (2017, 1985);
+-----+-----+
| title          | released_year |
+-----+-----+
| A Heartbreaking Work of Staggering Genius | 2001 |
| Caroline      | 2003 |
+-----+-----+
2 rows in set (0.00 sec)
```

## Joining Functions in SQL

Joining data in SQL means returning columns that have a specific relation to them.

- **cross join**

cross join returns rows from the table that were joined. The syntax is:

```
select <column name> from <table name>
```

```
cross join <table name> on <column name.id>=<column name.id>;
```

- **inner join**

inner join combine rows from the table that have matching values The syntax is:

```
select <column name> from <table name>
```

```
inner join <table name> on <column name.id>=<column name.id>;
```

For example:



```
[mysql] SELECT title  
[      -> FROM books  
[      -> INNER JOIN students;  
Empty set (0.00 sec)
```

- **left join**

left join returns records from left table and matching values from the right table. The syntax is:

```
select <column name> from <table name>
```

```
left join <table name> on <column name.id>=<column name.id>;
```

- **right join**

right join returns records from right table and matching values from the left table. The syntax is:

```
select <column name> from <table name>
```

```
right join <table name> on <column name.id>=<column name.id>;
```