Basic Syntax

In this chapter, we will go over the basic SQL syntax.

SQL statements are basically the 'commands' that you run against a specific database. Through the SQL statements, you are telling MySQL what you want it to do, for example, if you wanted to get the `username` of all of your users stored in the `users` table, you would run the following SQL statement:

```
SELECT username FROM users;
```

Rundown of the statement:

* `SELECT`: First, we specify the `SELECT` keyword, which indicates that we want to select some data from the database. Other popular keywords are: `INSERT`, `UPDATE` and `DELETE`.
* `username`: Then we specify which column we want to select.
* `FROM users`: After that, we specify the table that we want to select the data from using the `FROM` keyword.
* The semicolon `;` is highly recommended to put at the end. Standard SQL syntax requires it, but some "Database Management Systems' (DBMS)" are tolerant about it, but it's not worth the risk.

If you run the above statement, you will get no results as the new `users` table that we've just created is empty.

As a good practice, all SQL keywords should be with uppercase, however, it would work just fine if you use lower case as well.
Let's go ahead and cover the basic operations next.

INSERT

To add data to your database, you would use the `INSERT` statement.

Let's use the table that we created in the last chapter and insert 1 user into our `users` table:

```sql
INSERT INTO users (username, email, active)
VALUES ('bobby', 'bobby@bobbyiliev.com', true);
```

Rundown of the insert statement:

- `INSERT INTO`: first, we specify the `INSERT INTO` keyword, which tells MySQL that we want to insert data a table.
- `users (username, email, active)`: then, we specify the table name `users` and the columns that we want to insert data into.
- `VALUES`: then, we specify the values that we want to insert in. The order of attributes is the same as in `users (...)`.

SELECT

Once we've inserted that user, let's go ahead and retrieve the information.

To retrieve information from your database, you could use the `SELECT` statement:

```
SELECT * FROM users;
```

Output:

```
+----+----------+-------+----------+--------+---------------+
| id | username | about | birthday | active | email         |
+----+----------+-------+----------+--------+---------------+
|  1 | bobby    | NULL  | NULL     |      1 | bobby@b...com |
+----+----------+-------+----------+--------+---------------+
```

We specify `*` right after the `SELECT` keyword, this means that we want to get all of the columns from the `users` table.

If we wanted to retrieve only the `username` and the `email` columns instead, we would change the statement to:

```
SELECT username, email FROM users;
```

This will return all of the users, but as of the time being we have only 1:

```
+----------+----------------------+
| username | email                |
+----------+----------------------+
| bobby    | bobby@bobbyiliev.com |
+----------+----------------------+
```

UPDATE

In order to modify data in your database, you could use the `UPDATE` statement.

The syntax would look like this:

```
UPDATE users SET username='bobbyiliev' WHERE id=1;
```

Rundown of the statement:

- `UPDATE users`: First, we specify the `UPDATE` keyword followed by the table that we want to update.
- `SET username='bobbyiliev'`: Then we specify the columns that we want to update and the new value that we want to set.
- `WHERE id=1`: Finally, by using the `WHERE` clause, we specify which user should be updated. In our case it is the user with ID 1.

NOTE: If we don't specify a `WHERE` clause, all of the entries inside the `users` table would be updated, and all users would have the `username` set to `bobbyiliev`. You need to be careful when you use the `UPDATE` statement without a `WHERE` clause, as every single row will be updated.

We are going to cover `WHERE` more in-depth in the next few chapters.

DELETE

As the name suggests, the `DELETE` statement would remove data from your database.

The syntax is as follows:

```
DELETE FROM users WHERE id=1;
```

Similar to the `UPDATE` statement, if you don't specify a `WHERE` clause, all of the entries from the table will be affected, meaning that all of your users will be deleted.

Comments

In case that you are writing a larger SQL script, it might be helpful to add some comments so that later on, when you come back to the script, you would know what each line does.

As with all programming languages, you can add comments in SQL as well.

There are two types of comments:

- Inline comments:

To do so, you just need to add `--` before the text that you want to comment out:

```sql
SELECT * FROM users; -- Get all users
```

- Multiple-line comments:

Similar to some other programming languages in order to comment multiple lines, you could wrap the text in `/* */` as follows:

```sql
/*
Get all of the users
from your database
*/
SELECT * FROM users;
```

You could write that in a `.sql` file and then run it later on, or execute the few lines directly.

Conclusion

Those were some of the most common basic SQL statements.

In the next chapters, we are going to go over each of the above statements more in-depth.