Sorting with ORDER and GROUP BY

In the last chapter, you've learned how to use the `SELECT` statement with the `WHERE` clause and filter the result set based on some conditions.

More often than not, you would want to order the results in a specific way based on a particular column. For example, you might want to order the users alphabetically, based on their username.

In this chapter, you will learn how to use the `ORDER BY` and `GROUP BY` clauses.

ORDER BY

The main thing that you need to keep in mind when using `ORDER BY` is to specify the column or columns you want to order by. In case that you want to specify multiple columns to order by, you need to separate each column with a comma.

If we were to run the following statement without providing an `ORDER BY` clause:

```sql
SELECT id, username FROM users;
```

We will get the following output:

```
+----+----------+
| id | username |
+----+----------+
|  2 | bobby    |
|  3 | devdojo  |
|  4 | tony     |
|  5 | bobby    |
|  6 | devdojo  |
|  7 | tony     |
+----+----------+
```

As you can see, the result set is sorted by the primary key, which in our case is the `id` of each user. If we wanted to sort the output by `username`, we would run the following query:

```sql
SELECT id, username FROM users ORDER BY username;
```

> Note: The `ORDER BY` statement is followed by the column's name that we want to order by.

The output, in this case, will be:

```
+----+----------+
| id | username |
+----+----------+
|  2 | bobby    |
|  5 | bobby    |
|  3 | devdojo  |
|  6 | devdojo  |
|  4 | tony     |
|  7 | tony     |
+----+----------+
```

> Note: You can use `ORDER BY` with and without specifying a `WHERE` clause. If you've used a `WHERE` clause, you need to put the `ORDER BY` clause after the `WHERE` clause.

The default sorting is ascending and is specified with the `ASC` keyword, and you don't need to add it explicitly, but if you want to sort by descending order, you need to use the `DESC` keyword.

If we use the query above and add `DESC` at the end as follows:

```sql
SELECT id, username FROM users ORDER BY username DESC;
```

We will see the following output:

```
+----+----------+
| id | username |
+----+----------+
|  4 | tony     |
|  7 | tony     |
|  3 | devdojo  |
|  6 | devdojo  |
|  2 | bobby    |
|  5 | bobby    |
+----+----------+
```

As you can see, we've got the same list of users sorted alphabetically but in reverse order.

GROUP BY

The `GROUP BY` statement allows you to use a function like `COUNT`, `MIN`, `MAX` etc., with multiple columns.

For example, let's say that we wanted to get all of the counts of all users grouped by username.

In our case, we have two users with username `bobby`, two users with username `tony`, and two users with username `devdojo`. This represented in an SQL statement would look like this:

```
SELECT COUNT(username), username FROM users GROUP BY username;
```

The output, in this case, would be:

```
+-----------------+----------+
| COUNT(username) | username |
+-----------------+----------+
|               2 | bobby    |
|               2 | devdojo  |
|               2 | tony     |
+-----------------+----------+
```

The `GROUP BY` statement grouped the identical usernames. Then it ran a `COUNT` on each of `bobby`, `tony` and `devdojo`.

The main thing to remember here is that the `GROUP BY` should be added after the `FROM` clause and after the `WHERE` clause.

HAVING Clause

The `HAVING` clause allows you to filter out the results on the groups formed by the `GROUP BY` clause.

For example, let's say that we wanted to get all usernames that are duplicates, i.e., all the usernames present in more than one table record.

In our case, we have two users with username `bobby`, two users with username `tony`, and two users with username `devdojo`. This represented in an SQL statement would look like this:

```
SELECT COUNT(username), username
FROM users
GROUP BY username
HAVING COUNT(username) > 1;
```

The output, in this case, would be:

```
+-----------------+----------+
| COUNT(username) | username |
+-----------------+----------+
|               2 | bobby    |
|               2 | devdojo  |
|               2 | tony     |
+-----------------+----------+
```

The `GROUP BY` clause grouped the identical usernames, calculated their counts and filtered out the groups using the `HAVING` clause.

> **NOTE** :- *The WHERE clause places conditions on the selected columns, whereas the HAVING clause places conditions on groups created by the GROUP BY clause.*