SQL Statements

Now that we understand how to use the `mysql` utility and create databases and tables, let us look at some of the essential SQL statements and their uses.

INSERT Statement

The INSERT statement is used to add new records to a given table. The statement following the below syntax:

Code: sql

```
INSERT INTO table_name VALUES (column1_value, column2_value, column3_value, ...);
```

The syntax above requires the user to fill in values for all the columns present in the table.

SQL Statements

```
mysql> INSERT INTO logins VALUES(1, 'admin', 'p@ssw0rd', '2020-07-02');

Query OK, 1 row affected (0.00 sec)
```

The example above shows how to add a new login to the logins table, with appropriate values for each column. However, we can skip filling columns with default values, such as `id` and `date_of_joining`. This can be done by specifying the column names to insert values into a table selectively:

Code: sql

```
INSERT INTO table_name(column2, column3, ...) VALUES (column2_value, column3_value, ...);
```

Note: skipping columns with the 'NOT NULL' constraint will result in an error, as it is a required value.

We can do the same to insert values into the `logins` table:

SQL Statements

```
mysql> INSERT INTO logins(username, password) VALUES('administrator', 'adm1n_p@ss');

Query OK, 1 row affected (0.00 sec)
```

We inserted a username-password pair in the example above while skipping the `id` and `date_of_joining` columns.

Note: The examples insert cleartext passwords into the table, for demonstration only. This is a bad practice, as passwords should always be hashed/encrypted before storage.

We can also insert multiple records at once by separating them with a comma:

SQL Statements

```
mysql> INSERT INTO logins(username, password) VALUES ('john', 'john123!'), ('tom', 'tom123!');

Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

The query above inserted two new records at once.

SELECT Statement

Now that we have inserted data into tables let us see how to retrieve data with the SELECT statement. This statement can also be used for many other purposes, which we will come across later. The general syntax to view the entire table is as follows:

Code: sql

```
SELECT * FROM table_name;
```

The asterisk symbol (*) acts as a wildcard and selects all the columns. The `FROM` keyword is used to denote the table to select from. It is possible to view data present in specific columns as well:

Code: sql

```
SELECT column1, column2 FROM table_name;
```

The query above will select data present in column1 and column2 only.

SQL Statements

```
mysql> SELECT * FROM logins;

+----+--------------+------------+---------------------+
| id | username     | password   | date_of_joining     |
```


My Workstation

OFFLINE

Start Instance

∞ / 1 spawns left

```
|  1 | admin         | p@ssw0rd   | 2020-07-02 00:00:00 |
|  2 | administrator | adm1n_p@ss | 2020-07-02 11:30:50 |
|  3 | john          | john123!   | 2020-07-02 11:47:16 |
|  4 | tom           | tom123!    | 2020-07-02 11:47:16 |
+----+---------------+------------+---------------------+
4 rows in set (0.00 sec)


mysql> SELECT username,password FROM logins;

+---------------+------------+
| username      | password   |
+---------------+------------+
| admin         | p@ssw0rd   |
| administrator | adm1n_p@ss |
| john          | john123!   |
| tom           | tom123!    |
+---------------+------------+
4 rows in set (0.00 sec)
```

The first query in the example above looks at all records present in the logins table. We can see the four records which were entered before. The second query selects just the username and password columns while skipping the other two.

DROP Statement

We can use DROP to remove tables and databases from the server.

SQL Statements

```
mysql> DROP TABLE logins;

Query OK, 0 rows affected (0.01 sec)


mysql> SHOW TABLES;

Empty set (0.00 sec)
```

As we can see, the table was removed entirely.

> The 'DROP' statement will permanently and completely delete the table with no confirmation, so it should be used with caution.

ALTER Statement

Finally, We can use ALTER to change the name of any table and any of its fields or to delete or add a new column to an existing table. The below example adds a new column `newColumn` to the `logins` table using `ADD`:

SQL Statements

```
mysql> ALTER TABLE logins ADD newColumn INT;

Query OK, 0 rows affected (0.01 sec)
```

To rename a column, we can use `RENAME COLUMN`:

SQL Statements

```
mysql> ALTER TABLE logins RENAME COLUMN newColumn TO newerColumn;

Query OK, 0 rows affected (0.01 sec)
```

We can also change a column's datatype with `MODIFY`:

SQL Statements

```
mysql> ALTER TABLE logins MODIFY newerColumn DATE;

Query OK, 0 rows affected (0.01 sec)
```

Finally, we can drop a column using `DROP`:

SQL Statements

```
mysql> ALTER TABLE logins DROP newerColumn;

Query OK, 0 rows affected (0.01 sec)
```

We can use any of the above statements with any existing table, as long as we have enough privileges to do so.

UPDATE Statement

While `ALTER` is used to change a table's properties, the UPDATE statement can be used to update specific records within a table, based on certain conditions. Its general syntax is:

Code: sql

```
UPDATE table_name SET column1=newvalue1, column2=newvalue2, ... WHERE <condition>;
```

We specify the table name, each column and its new value, and the condition for updating records. Let us look at an example:

SQL Statements

```
mysql> UPDATE logins SET password = 'change_password' WHERE id > 1;

Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

```
mysql> SELECT * FROM logins;

+----+---------------+----------------+---------------------+
| id | username      | password       | date_of_joining     |
+----+---------------+----------------+---------------------+
|  1 | admin         | p@ssw0rd       | 2020-07-02 00:00:00 |
|  2 | administrator | change_password | 2020-07-02 11:30:50 |
|  3 | john          | change_password | 2020-07-02 11:47:16 |
|  4 | tom           | change_password | 2020-07-02 11:47:16 |
+----+---------------+----------------+---------------------+
4 rows in set (0.00 sec)
```

The query above updated all passwords in all records where the id was more significant than 1.

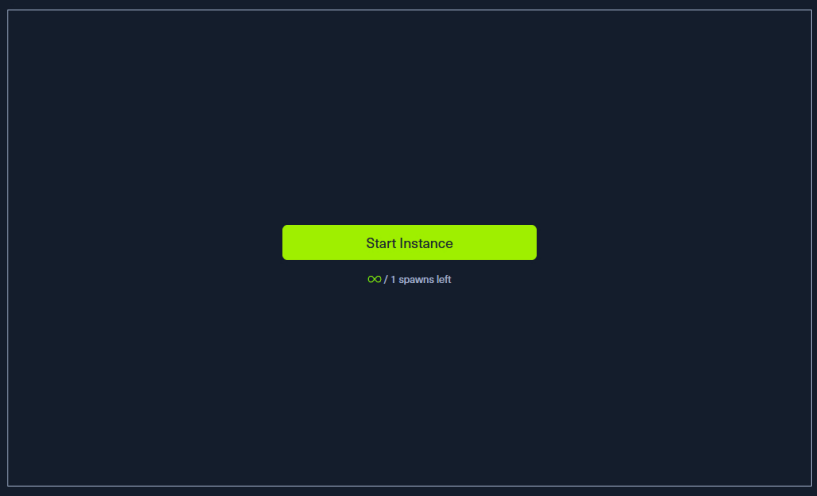
Note: we have to specify the 'WHERE' clause with UPDATE, in order to specify which records get updated. The 'WHERE' clause will be discussed next.

Connect to Pwnbox
Your own web-based Parrot Linux instance to play our labs.

Pwnbox Location

UK 159ms

Terminate Pwnbox to switch location



Waiting to start...

Enable step-by-step solutions for all questions

Questions

Answer the question(s) below to complete this Section and earn cubes!

Cheat Sheet

Target(s): Click here to spawn the target system!

Authenticate to with user "root" and password "password"

+0 What is the department number for the 'Development' department?

d005

Submit Hint

← Previous Next →

Mark Complete & Next