Security and injection attacks

So far, we have been using Python’s string formatting method to insert parameters like table and column names into the c.execute() functions. This is fine if we just want to use the database for ourselves. However, this leaves our database vulnerable to injection attacks. For example, if our database would be part of a web application, it would allow hackers to directly communicate with the database in order to bypass login and password verification and steal data.  
In order to prevent this, it is recommended to use ? place holders in the SQLite commands instead of the % formatting expression or the .format() method, which we have been using in this tutorial.  
For example, instead of using

*# 5) Check if a certain ID exists and print its column contents*

c**.**execute("SELECT \* FROM {tn} WHERE {idf}={my\_id}"**.**\

format(tn**=**table\_name, cn**=**column\_2, idf**=**id\_column, my\_id**=**some\_id))

in the [Querying the database - Selecting rows](https://sebastianraschka.com/Articles/2014_sqlite_in_python_tutorial.html#querying) section above, we would want to use the ? placeholder for the queried column value and include the variable(s) (here: 123456), which we want to insert, as tuple at the end of the c.execute() string.

*# 5) Check if a certain ID exists and print its column contents*

c**.**execute("SELECT \* FROM {tn} WHERE {idf}=?"**.**\

format(tn**=**table\_name, cn**=**column\_2, idf**=**id\_column), (123456,))

However, the problem with this approach is that it would only work for values, not for column or table names. So what are we supposed to do with the rest of the string if we want to protect ourselves from injection attacks? The easy solution would be to refrain from using variables in SQLite queries whenever possible, and if it cannot be avoided, we would want to use a function that strips all non-alphanumerical characters from the stored content of the variable, e.g.,

**def** **clean\_name**(some\_var):

**return** ''**.**join(char **for** char **in** some\_var **if** char**.**isalnum())