# **Database Lab 3 - Introduction to Python for accessing SQLite**

**Objectives**

* To introduce some basic features of the Python API for accessing Databases
* To produce some small Python scripts that access SQLite database
* To provide confidence to learn more Python and further DB API features

**Background Knowledge**

By now you should have completed lab 1 and lab 2 and have some experience of basic SQL commands such as “CREATE”, “SELECT” and “INSERT”. This lab, you will learn how to query data in SQLite from Python.

We recommend you use Repl.it to practise these queries. However, if you wish to complete this worksheet on your own PC you will have to install Python version 3 and the sqlite3 python module. Details of how to install these pieces of software are beyond the scope of this tutorial, but can be easily found from many internet sites. A number of "dynamic tutorials" that demonstrate the installation of this software are provided in the software section of this website.

Note that this tutorial is not intended to be exhaustive, it is a simple introduction. You should ensure that you access the other resources available on the internet and in relevant textbooks to find out more.

**In this lab you will be,**

1. Creating a Python program on Repl

Click new repl button on the top menu and then search Python. The main.py will be created in your repl repository.

2. The first script

The following is the typical "Hello World" program for Python. Use an editor to type it in as shown below;

print (“Hello World!”)

You should see a response text "Hello World”.

3. Connecting to a sqlite3 database

We are going to be using an example database in this session. Since we have learnt to use a Python script, we may as well use Python to set up the example database. Type (or cut-and-paste) the following the main.py file.

import sqlite3

conn = sqlite3.connect('store')

conn.execute("CREATE TABLE 'pet' (name VARCHAR(20), owner VARCHAR(20), species VARCHAR(20), sex CHAR(1), checkups SMALLINT UNSIGNED, birth DATE, death DATE)")

When you run the script, store.db file will be automatically created. You will not see any data in the database just yet.

Before moving on, take some time to have a look again at the script you have just run. Go through the script and put a comment above each statement that describes what you think the statement is doing.

Try running the script more than one time. What did you get?

Now let’ try this script:

import sqlite3

conn = sqlite3.connect('store')

print ("Database has been created")

conn.execute("DROP TABLE IF EXISTS pet")

conn.execute("CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20), species VARCHAR(20), sex CHAR(1), checkups SMALLINT UNSIGNED, birth DATE, death DATE)")

print ("Table created successfully")

4. Inserting Data

Inserting data into a table in the database is not the simplest of the things you might want to do but is clearly fundamental to any database work with Python.

A new record is inserted into a table using the INSERT query. An example is the following:

"INSERT INTO pet VALUES (1, 'Fluffy', 'Alice', 'cat', 'f', 5, '2001-02-04', null)"

It is possible to specify which fields data will be inserted into too. Default values will be supplied in any fields not used, as long as these default values do not contravene the data description of the record.

"INSERT INTO pet (name, owner, species, birth) VALUES ('Fluffy', 'Alice', 'cat', '2001-02-04')"

Now let’s use this knowledge to insert some data. Consider the following Python script:

conn.execute("INSERT INTO pet (name,owner,species,sex,checkups,birth,death)VALUES \

('Fluffy','Harold','cat','f',5,'2001-02-04','')")

conn.execute("INSERT INTO pet (name,owner,species,sex,checkups,birth,death)VALUES \

('Claws','Gwen','cat','m',2,'2000-03-17','')")

conn.commit()

print("Records created successfully")

print("Total number of rows created :", conn.total\_changes)

What do you think will happen when the script is executed?

5. Reporting on Data

As well as being able to enter data, we need to be able to retrieve data from the table. This is relatively simple for single value queries, as illustrated in the following script:

You can see that a query that results in a list being returned. The data in the list in arranged in the same order as the columns of the table created earlier.

cursor = conn.execute("SELECT name,owner,species,sex,checkups,birth,death from pet")

for row in cursor:

print("name = ", row[0])

print("owner = ", row[1])

print("species = ", row[2])

print("sex = ", row[3])

print("checkups = ", row[4])

print("birth = ", row[5])

print("death = ", row[6], "\n")

**6. Updating data**

Now that you have seen how it is done, look up the UPDATE operation in the UPDATE syntax section of the SQLite Manual and provide a Python script that allows a death date to be entered for an animal identified by its name and owner's name

7. Further Work

We have really only started to look at how to use SQLite through Python sqlite3 module. There is much more to learn, and much more experience to be gained. You should ensure that you access the tutorials and books available to increase your Python skills and pick up good Python programming tips prior to tackling later tutorials. You will find that the practice you will get doing further lab work will help you refine the skills you have learnt.

We have been using the sqlite3 module in this tutorial.