Week 9 Seminar: SQLite & Python

Objectives of computer seminar

1. Create a database and table using Python and SQLite
2. Add records
3. Search for data
4. Optional Exercises: Delete records, Update records

Create a database and table using Python and SQLite

We will create a database to hold contact information: first name, last name, email, phone.

1. To create a new database *contact.db* the first steps are:
    - import the sqlite3 library,
    - make a connection,
    - create a cursor object. The cursor object will be used to execute SQL queries.

```
import sqlite3                          # Import the sqlite3 library

db = sqlite3.connect("contact.db")      # Create Connection object
cursor = db.cursor()                    # Create Cursor object
```

2. Create a contacts table:

```
# Create a string to hold the SQL query
sql = """
      CREATE TABLE IF NOT EXISTS contacts (
      contact_id integer PRIMARY KEY,
      first_name text,
      last_name text,
      email text,
      phone text); """

cursor.execute(sql)          # Pass the string
db.commit()                  # Save change permanently
```

Notes on example:

1. Here we pass a string containing the SQL query including the CREATE TABLE statement to the `execute()` method of the `Cursor` object.
    - Wrap the SQL query in quotes. Use single, double, or triple quotes. However, triple quotes allow you to write multi-line queries.
2. We create five columns: *contact_id, first_name, last_name, email, phone*.
3. contact_id is assigned as the primary key.
4. Note: If we try to create a table that already exists it will produce an error. To check if the table doesn't already exist, we use IF NOT EXISTS with the CREATE TABLE statement.

5. Commit the changes (save permanently) by using the `commit()` function on the `Connection` object.

Add records

Insert some contact data into the *contacts* table.

```
sql = """
INSERT INTO contacts (first_name, last_name, email, phone)
VALUES ('Boris', 'Johnson', 'bj@number10.com', '12345678');
"""
cursor.execute(sql)                    # Pass the string variable
db.commit()                            # Save permanently
```

Notes on example:

- Specify the name of the table we want to add values into.
- Specify a list of all the columns in the table. While this list is optional, it's good practice to include it. Then follow with a list of values we want to include.
- If we don't include all the column names, we have to include a value for each column.
- Here we create a string containing the SQL statement and pass that string to the cursor.execute() command
- Remember to wrap the SQL query in quotes (single, double, or triple). Triple quotes allow you to write multi-line queries.
- A value is not required for contact_id as this column was defined as "integer PRIMARY KEY". The values for this will be generated by SQLite automatically.
- To insert rows we use the cursor object to execute the query.

Run a query to check that the data has been added:

```
sql= "Select * from contacts"
cursor.execute(sql)
print(cursor.fetchall())
```

- Add more contacts using the methods shown in the earlier example (create a SQL string and pass the string to cursor.execute(). Or experiment with method 2 and 3 below.

- Method 2 - Pass the data directly to the cursor.execute() command.
- Method 3 - qmark parameters - Replace all the values with question marks (?) and add an additional parameter that will contain the values to be added. The parameter passed should be a tuple.

Note: The **DROP TABLE** statement can be used to delete the contacts table. Useful if you have multiple records for the same contact and want to start the table again.

```
sql = """ DROP TABLE contacts """
cursor.execute(sql)
db.commit()
```

Search for data

1. We have already used the following to select all data from the contacts table.

```
sql= "Select * from contacts"
cursor.execute(sql)
print(cursor.fetchall())
```

- The * symbol is known as a wildcard.
- `fetchall()` - returns all records that were selected by the SELECT command.
- `fetchone()` - returns one record that was selected by the SELECT command.

2. Here is an example using fetchone():

```
sql = "SELECT * FROM contacts WHERE contact_id = 1"
cursor.execute(sql)
print(cursor.fetchone())     # Fetchone useful here
```

Closing the database

db.close() to close the database connection when you have finished.

Additional Exercises (optional)

1. Using SELECT
   a) To show only certain columns state the columns instead of using the wildcard (*).

```
sql = "SELECT first_name, last_name FROM contacts;"
cursor.execute(sql)
print(cursor.fetchall())
```

   b) To show films in order we use the statement ORDER BY.

```
sql = """
SELECT first_name, last_name FROM contacts ORDER
BY last_name Asc; """

cursor.execute(sql)
print(cursor.fetchall())
```

2. Add the code to delete one of the contacts. Test that this has worked.

3. Add the code to amend one of the phone numbers. Test this has worked.

4. Ask the user to input a particular first name and then print out the contact details for the person(s) with that first name.