# Introduction to Databases with Python and SQLite

### Aim

* Experience how data can be stored in a database.
* Be able to Create, and alter the structure of the database using DDL
* Be able to Add, Edit, and retrieve data using SQL
* Be able to visualise the structure of the data

## Tasks

You are going to create a database using sqlite and DDL (Data Definition Language). When on a college PC you can use the online sqlite viewer (using <https://inloop.github.io/sqlite-viewer/>) to view the database you create. You could also use the DB Browser for SQLite by navigating to the *‘Z:\ICT\DB Browser for SQL lite’* folder. On your own PC you can use the DB Browser for SQLite (from <https://sqlitebrowser.org/>). The database you will create will hold some Student Information, Exam information and exam Results.

### Task1 – Create Form using Tkinter for Python

1. Create a new windows program using tkinter (see Task 2 if you are unsure how to do this). For ease you can create a new button for each of the following tasks. So by the end you will have a form with lots of buttons on it…
2. Name the buttons something sensible (e.g. btnCreateDataBase).. You should also alter the display text so each button can be identified when the program runs.
3. To work with sqlite databases you need to import the sqlite3 library:

import sqlite3

from sqlite3 import Error

### Task 2 - Create a Database

The code for creating the database is below… It uses the connect function which creates the database if it does not already exist. Note that it will create the database file (Test.db) in the same folder as your Python code.

def createdatabase():

 try:

 sqlite3.connect("Test.db")

 messagebox.showinfo("Info","Database created successfully")

 except Error as e:

 messagebox.showinfo("Error","Database not created: "+str(e))

### Make your first button call the above procedure when it is clicked.

Have a go at this – if you get stuck, here’s some sample code:

import sqlite3

from sqlite3 import Error

import tkinter as tk

from tkinter import messagebox

# Event handlers

def createdatabase():

 try:

 sqlite3.connect("Test.db")

 tk.messagebox.showinfo("Info","Database created successfully")

 except Error as e:

 tk.messagebox.showinfo("Error","Database not created: "+str(e))

# Set up UI

window = tk.Tk()

btnCreateDatabase = tk.Button(text = "Create Database", command = createdatabase)

btnCreateDatabase.pack()

# Display UI and listen for events

window.mainloop()

### Info - Opening and Closing the connection to the Database

For information, before you execute any DDL (e.g. create tables, add records) you need to create a database connection. You use the connection to run DDL commands, you also need it to close the database (which you must always do, just like you must always close a file when you’ve finished viewing/updating in your code). To do this you use a variable that stores connection details (this is the *myconnection* variable below).

Note the myconnection variable is returned from the function so it can be used to run DDL commands and close the database.

Here’s the opendatabase code.

def opendatabase():

 myconnection = None

 try:

 myconnection = sqlite3.connect("Test.db")

 messagebox.showinfo("Info","Database opened successfully")

 except Error as e:

 messagebox.showinfo("Error","Database not opened: "+str(e))

 return myconnection

Here’s how you’d use the opendatabase function to run DDL

def onclick():

 myconnection = opendatabase()

 #Insert DDL code that uses the connection (myconnection)

 myconnection.close()

### Task 3 – Creating tables

Now let’s start using DDL proper! We will be creating 3 tables: *Students*, *Exams*, and *Results*. I’ll give you the code for students and specify the other 2 tables…

Note that the three quotes means you can split literal text over more than one line. This makes the DDL a lot easier to read.

def createstudenttable():

 # put the DDL (which will create the Students table) into a string variable

 DDLstr = """CREATE TABLE IF NOT EXISTS Students (

 Username text PRIMARY KEY,

 FirstName text,

 Surname text)"""

 try:

 # get a database connection object

 myconnection = opendatabase()

 # use the connection object to create a command object

 mycommand = myconnection.cursor()

 # use the command object to execute the DDL

 mycommand.execute(DDLstr)

 # use the connection object to close the database

 myconnection.close()

 messagebox.showinfo("Info","Students Table created")

 except Error as e:

 messagebox.showinfo("Error","Students table not created: "+str(e))

Really check that you understand this.

Now create the Exams Table, which will hold information about the different exam each subject has

|  |  |
| --- | --- |
| Field | Properties |
| ExamCode | text , Primary Key |
| ExamTitle | Text |
| Subject | Text |

Now create the Results Table

|  |  |
| --- | --- |
| Field | Properties |
| ResultID | text , Primary Key |
| UserName | Text |
| ExamCode | Text |
| Score | Integer |

We will be adding extra fields to the 3 tables later

Open the sqlite database file and look at the tables that have been created… does everythong look correct?

### Task 4 Adding Fields to a table

Here is an example that adds a field to an existing table:

def createfield():

 DDLstr = """ALTER TABLE Exams ADD COLUMN Weight real"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(DDLstr)

 myconnection.close()

 messagebox.showinfo("Info","Weight column added")

 except Error as e:

 messagebox.showinfo("Error","Weight column not added: "+str(e))

Now add the following:

 # DDLstr = "ALTER TABLE Results ADD COLUMN RawScore real"

 # Add MaxScore field to the Exams Table with type integer

 # Add a DateofBirth Field to the Students Table

 # Add a Gender Field to Students Table

 # Add ExamDate to Results (just storing the year as an integer)

### Task 5 Adding data to a table

Let’s start with adding some hard coded data (you will never do this in practice but that way you can see the syntax…

def insertonerecord():

 SQLstr = """INSERT INTO Students(Username, FirstName, Surname)

 VALUES ('BB','Robert','theBuilder')"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(SQLstr)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Record inserted")

 except Error as e:

 messagebox.showinfo("Error","Record not inserted: "+str(e))

Make sure you understand the syntax…

* Why are single quotes needed above?

Now we will do the same thing again BUT with data stored in variables

def insertanotherstudentrecord():

 uName = "WB"

 fName = "Wendy"

 sName = "theBuilder"

 SQLstr = f"""INSERT INTO Students(Username, FirstName, Surname) VALUES ('{uName}','{fName}','{sName}')"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(SQLstr)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Record inserted")

 except Error as e:

 messagebox.showinfo("Error","Record not inserted: "+str(e))

Alter the Procedure to add the following data to the **EXAMS** Table. The column headings are the exact field names..

| **ExamCode** | **ExamTitle** | **Subject** | **Weight** | **MaxScore** |
| --- | --- | --- | --- | --- |
| COMP999 | Geekdom | Computing | 0.6 | 100 |

Add Textboxes to your UI to allow you to enter data into the **Students** and **Exams** Tables.

### Task 6 Adding data to the database from Text files.

It is often useful to have sets of test data stored as text files.

Here is an example for the Student database…

from tkinter.filedialog import askopenfilename

def adddatafromfile(tablename):

 sql = "INSERT INTO " + tablename + " VALUES"

 if tablename == "Students":

 sql += " (?,?,?,?,?)"

 elif tablename == "Exams":

 sql += " (?,?,?,?,?)"

 elif tablename == "Results":

 sql += " (?,?,?,?,?,?)"

 else:

 messagebox.showerror("Error",f"ERROR creating table {tablename}")

 return

 records = []

 fname = askopenfilename()

 try:

 fhandle = open(fname, "r")

 rows = fhandle.read().splitlines()

 for row in rows:

 values = row.split("\t")

 record = tuple(values)

 records.append(record)

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.executemany(sql, records)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info",f"{tablename} records inserted from file")

 except Error as e:

 messagebox.showerror("Error",f"ERROR when inserting {tablename} records from file:\n"+str(e))

 Add to the above code to insert the data from files for the Exams and Results Tables…

Sample Python Solution

from tkinter import \*

from tkinter import messagebox

from tkinter.filedialog import askopenfilename

import sqlite3

from sqlite3 import Error

win = Tk()

win.geometry("1200x600")

username\_var=StringVar()

firstname\_var=StringVar()

surname\_var=StringVar()

dob\_var=StringVar()

gender\_var=StringVar()

def createdatabase():

 try:

 sqlite3.connect("Test.db")

 messagebox.showinfo("Info","Database created successfully")

 except Error as e:

 messagebox.showinfo("Error","Database not created: "+str(e))

def opendatabase():

 myconnection = None

 try:

 myconnection = sqlite3.connect("Test.db")

 #messagebox.showinfo("Info","Database opened successfully")

 except Error as e:

 messagebox.showinfo("Error","Database not opened: "+str(e))

 return myconnection

def closedatabase(connection):

 try:

 connection.close()

 messagebox.showinfo("Info","Database closed successfully")

 except Error as e:

 messagebox.showinfo("Error","Database not closed: "+str(e))

def onclick():

 myconnection = opendatabase()

 #Insert DDL code that uses the connection (myconnection)

 myconnection.close()

def createstudenttable():

 # put the DDL (which will create the Students table) into a string variable

 DDLstr = """CREATE TABLE IF NOT EXISTS Students (

 Username text PRIMARY KEY,

 FirstName text,

 Surname text)"""

 try:

 # get a database connection object

 myconnection = opendatabase()

 # use the connection object to create a command object

 mycommand = myconnection.cursor()

 # use the command object to execute the DDL

 mycommand.execute(DDLstr)

 # use the connection object to close the database

 myconnection.close()

 messagebox.showinfo("Info","Students Table created")

 except Error as e:

 messagebox.showinfo("Error","Students table not created: "+str(e))

def createexamstable():

 DDLstr = """CREATE TABLE IF NOT EXISTS Exams (

 ExamCode text PRIMARY KEY,

 ExamTitle text,

 Subject text)"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(DDLstr)

 myconnection.close()

 messagebox.showinfo("Info","Exams Table created")

 except Error as e:

 messagebox.showinfo("Error","Exams table not created: "+str(e))

def createresultstable():

 DDLstr = """CREATE TABLE IF NOT EXISTS Results (

 ResultID text PRIMARY KEY,

 UserName text,

 ExamCode text,

 Score integer)"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(DDLstr)

 myconnection.close()

 messagebox.showinfo("Info","Results Table created")

 except Error as e:

 messagebox.showinfo("Error","Results table not created: "+str(e))

def createfield():

 DDLstr = """ALTER TABLE Exams ADD COLUMN Weight real"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(DDLstr)

 myconnection.close()

 messagebox.showinfo("Info","Weight column added")

 except Error as e:

 messagebox.showinfo("Error","Weight column not added: "+str(e))

def runSQL(sql):

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(sql)

 myconnection.close()

 messagebox.showinfo("Info","SQL executed successfully")

 except Error as e:

 messagebox.showinfo("Error","SQL '" +sql+ "' failed: "+str(e))

def createextrafields():

 runSQL("""ALTER TABLE Results ADD COLUMN RawScore real""")

 runSQL("""ALTER TABLE Exams ADD COLUMN MaxScore integer""")

 runSQL("""ALTER TABLE Students ADD COLUMN DateOfBirth text""")

 runSQL("""ALTER TABLE Students ADD COLUMN Gender text""")

 runSQL("""ALTER TABLE Results ADD COLUMN ExamDate integer""")

def insertonerecord():

 SQLstr = """INSERT INTO Students(Username, FirstName, Surname)

 VALUES ('BB','Robert','theBuilder')"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(SQLstr)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Record inserted")

 except Error as e:

 messagebox.showinfo("Error","Record not inserted: "+str(e))

def insertanotherstudentrecord():

 uName = "WB"

 fName = "Wendy"

 sName = "theBuilder"

 SQLstr = f"""INSERT INTO Students(Username, FirstName, Surname) VALUES ('{uName}','{fName}','{sName}')"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(SQLstr)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Record inserted")

 except Error as e:

 messagebox.showinfo("Error","Record not inserted: "+str(e))

def insertexamrecord():

 SQLstr = """INSERT INTO Exams(ExamCode, ExamTitle, Subject, Weight, MaxScore)

 VALUES ('COMP999','Geekdom','Computing', 0.6, 100)"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(SQLstr)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Exam record inserted")

 except Error as e:

 messagebox.showinfo("Error","Exam record not inserted: "+str(e))

def addstudent():

 bIns = False

 SQLstr = f"""INSERT INTO Students(Username, FirstName, Surname, DateOfBirth, Gender)

 VALUES ('{username\_var.get()}','{firstname\_var.get()}','{surname\_var.get()}','{dob\_var.get()}','{gender\_var.get()}')"""

 try:

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.execute(SQLstr)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Student record inserted")

 bIns = True

 username\_var.set("")

 firstname\_var.set("")

 surname\_var.set("")

 dob\_var.set("")

 gender\_var.set("")

 except Error as e:

 if bIns:

 messagebox.showinfo("Error","Failed to reset fields: "+str(e))

 else:

 messagebox.showinfo("Error","Student Record not inserted: "+str(e))

def addstudentsfromfile():

 sql = "INSERT INTO Students VALUES (?,?,?,?,?)"

 # create a list to add tuples

 records = []

 # get file

 fname = askopenfilename()

 try:

 fhandle = open(fname, "r")

 # create a list where each element is a row of the file

 rows = fhandle.read().splitlines()

 # loop through the rows

 for row in rows:

 # create a list of 'values' from the current row so can use the 'executemany' method later.

 # assume that values in the fiie are seperated by tab characters

 values = row.split("\t")

 # convert the list to a 'tuple'

 record = tuple(values)

 # and add it to the list of tuples

 records.append(record)

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 # execute the sql passing in the list of tuples to insert into the table

 mycommand.executemany(sql, records)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info","Student records inserted from file")

 except Error as e:

 messagebox.showerror("Error","ERROR when inserting Student records from file:\n"+str(e))

def adddatafromfile(tablename):

 sql = "INSERT INTO " + tablename + " VALUES"

 if tablename == "Students":

 sql += " (?,?,?,?,?)"

 elif tablename == "Exams":

 sql += " (?,?,?,?,?)"

 elif tablename == "Results":

 sql += " (?,?,?,?,?,?)"

 else:

 messagebox.showerror("Error",f"ERROR creating table {tablename}")

 return

 records = []

 fname = askopenfilename()

 try:

 fhandle = open(fname, "r")

 rows = fhandle.read().splitlines()

 for row in rows:

 values = row.split("\t")

 record = tuple(values)

 records.append(record)

 myconnection = opendatabase()

 mycommand = myconnection.cursor()

 mycommand.executemany(sql, records)

 myconnection.commit()

 myconnection.close()

 messagebox.showinfo("Info",f"{tablename} records inserted from file")

 except Error as e:

 messagebox.showerror("Error",f"ERROR when inserting {tablename} records from file:\n"+str(e))

Button(win, text="Create Database", height=5, width=25, command=lambda:createdatabase()).grid(row=0,column=0)

Button(win, text="Open Database", height=5, width=25, command=lambda:onclick()).grid(row=0,column=1)

Button(win, text="Create Students table", height=5, width=25, command=lambda:createstudenttable()).grid(row=0,column=2)

Button(win, text="Create Exams table", height=5, width=25, command=lambda:createexamstable()).grid(row=0,column=3)

Button(win, text="Create Results table", height=5, width=25, command=lambda:createresultstable()).grid(row=0,column=4)

Button(win, text="Create Field", height=5, width=25, command=lambda:createfield()).grid(row=1,column=0)

Button(win, text="Create Extra Fields", height=5, width=25, command=lambda:createextrafields()).grid(row=1,column=1)

Button(win, text="Insert record", height=5, width=25, command=lambda:insertonerecord()).grid(row=1,column=2)

Button(win, text="Insert Another Record", height=5, width=25, command=lambda:insertanotherstudentrecord()).grid(row=1,column=3)

Button(win, text="Insert An Exam Record", height=5, width=25, command=lambda:insertexamrecord()).grid(row=1,column=4)

Label(win, text="Username").grid(row=2,column=0)

Label(win, text="FirstName").grid(row=2,column=1)

Label(win, text="Surname").grid(row=2,column=2)

Label(win, text="DateOfBirth").grid(row=2,column=3)

Label(win, text="Gender").grid(row=2,column=4)

Entry(win, textvariable = username\_var).grid(row=3, column=0)

Entry(win, textvariable = firstname\_var).grid(row=3, column=1)

Entry(win, textvariable = surname\_var).grid(row=3, column=2)

Entry(win, textvariable = dob\_var).grid(row=3, column=3)

Entry(win, textvariable = gender\_var).grid(row=3, column=4)

Button(win, text="Insert Student Details", height=5, width=25, command=lambda:addstudent()).grid(row=4,column=0)

Button(win, text="Insert Students from file", height=5, width=25, command=lambda:addstudentsfromfile()).grid(row=4,column=1)

Button(win, text="Insert Exams from file", height=5, width=25, command=lambda:adddatafromfile('Exams')).grid(row=4,column=2)

Button(win, text="Insert Results from file", height=5, width=25, command=lambda:adddatafromfile('Results')).grid(row=4,column=3)

win.mainloop()