Experiment – 10: Database

Aim: To demonstrate creation of a database using python.

Theory:

SQLite is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. SQLite is the most widely deployed SQL database engine in the world. The source code for SQLite is in the public domain.

The database in SQLite can be created as follows

# Creating a Database and Running SELECT query

```
import sqlite3

#conn = sqlite3.connect('employee.db') #uncomment this statement when executing the script the
1st time to create the database. Thereby you can comment in case of updating the database.

conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute(''' CREATE TABLE employee (
                first text,
                last text,
                age integer
                )''')


c.execute("INSERT INTO employee VALUES ('RATAN', 'TATA', 83)")
c.execute("INSERT INTO employee VALUES ('WARREN', 'BUFFET', 90)")
c.execute("INSERT INTO employee VALUES ('MUKESH', 'AMBANI', 63)")
c.execute("INSERT INTO employee VALUES ('ANIL', 'AMBANI', 61)")
conn.commit()

c.execute("SELECT * FROM employee WHERE last = 'AMBANI'")

print(c.fetchall())

conn.close()
```

One can create the database by passing values as objects too. Sample for the same is as given below

# Create a file called employees and define class

```
class Employee:
    def __init__(self, first, last, pay):
        self.first = first
        self.last = last
        self.age = pay


import sqlite3
from employees import Employee

conn = sqlite3.connect(':memory:')

c = conn.cursor()

c.execute(''' CREATE TABLE employee (
        first text,
        last text,
        age integer
        )''')

emp1 = Employee('Harry','Potter','34')
emp2 = Employee('Ron','Weasely','32')

c.execute("INSERT INTO employee VALUES (?,?,?)",(emp1.first,emp1.last,emp1.age))
c.execute("INSERT INTO employee VALUES (?,?,?)",(emp2.first,emp2.last,emp2.age))
conn.commit()

c.execute("SELECT * FROM employee WHERE first = ?",('Harry',))

print(c.fetchall())


conn.close()
```

The database creation can be automated using the following code:

```
# Automate using Functions
import sqlite3
from employees import Employee

#conn = sqlite3.connect('employee.db')
conn = sqlite3.connect(':memory:')

c = conn.cursor()

def insert_emp(emp):
    with conn:
        c.execute("INSERT INTO employee VALUES
(?,?,?)",(emp.first,emp.last,emp.age))

def get_emps_by_name(lastname):
    c.execute("SELECT * FROM employee WHERE last = ?",(lastname,))
    return c.fetchall()

def update_age(emp, age):
    with conn:
        c.execute("""UPDATE employee SET age = ? WHERE first = ? AND last
= ?""", (age, emp.first, emp.last ))


def remove_emp(emp):
    with conn:
        c.execute("DELETE from employee WHERE first = ? AND last = ?",
(emp.first, emp.last))


c.execute(''' CREATE TABLE employee (
        first text,
        last text,
        age integer
        )''')

emp1 = Employee('Chadwick','Boseman','34')
emp2 = Employee('Heath','Ledger','32')
emp2 = Employee('Cillian','Murphy','32')

insert_emp(emp1)
insert_emp(emp2)

emp = get_emps_by_name('Boseman')
print(emp)

update_age(emp3,35)
emp = get_emps_by_name('Murphy')
print(emp)

remove_emp(emp2)
emp = get_emps_by_name('Ledger')
print(emp)
conn.close()
```

Conclusion:

Task for submission:

(Write comments for every statement of the program)

1. Write python program to create database called myclass. The database will contain Students First_name, Last_name, Roll_number and Address in the Object form. Define methods to perform different operations on database and demonstrate the same.

   (Implement using all the 3 methods specified in the writeup)