Spring 2023 CSE 480
Project 2: DBMS - CREATE, INSERT, SELECT

January 23rd, 2023

---

1 A Word of Warning

Make sure you read through the submission instructions thoroughly before you submit your project. Failure to comply with the submission instructions will result in a zero for this project.

Projects 3, 4, and 5 will build off of the code your develop in project 2. So I highly recommend that you do not leave this project to the last day. I also highly recommend that you comment your code thoroughly so you can remember how your code works in the future.

We will not be releasing reference solutions to projects, so if you do not do a project, or do it poorly, you will have to finish that project before starting the next project. Because of this, we will offer a resubmission of this project one week after the initial deadline. See the class syllabus for how resubmissions will be graded. Additionally, we will give more help in office hours after the project deadline to help you finish a previous project that you might not have completed properly.

2 Project Due Date

This project is due before February 9th at 11:59pm. The second-chance deadline for project 2 is February 16th before 11:59pm. Submission instructions for this project can be found at the end of this document.

3 Project Overview

This project will emulate the behaviour of the built-in python module, "sqlite3". Your module will be able to execute SQL statements corresponding to: creating tables, inserting rows, and selecting rows.

4 What You are Given

On D2L we have released starter code for project 2, as well as some test cases, and a script to run the test cases. We have also released sample code to tokenize inputs, which you can choose to use, or not to use. If you will use our tokenizing code, it must be incorporated into your main project file.

5 What You Need To Do

Each of our testcases will import your "project.py" code and expects to find a "connect" function with one parameter, which is a filename (specifically test.db). This parameter is not used in this project, and will be used in later projects when we look into persistence.

The connect function should return an object that has two methods: "execute" and "close". For each SQL statement in a test case, the test case will pass that SQL statement as a string into the "execute" method. The execute method should return an empty list, unless a select statement was executed that yielded rows. in that case, a list of tuples with each denoting a row, should be returned. The "close" method will not be used for this project, so it does not need to be implemented for this project.

There is no need to write any database output to any files. Persistence will be covered in a later project. Each test will be done on a fresh database.

Your program is allowed to use any built-in modules of python, with the exception of sqlite3. sqlite3 is the reference implementation for this project, your output will be compared against it for correctness. Importing the sqlite3 module in project.py will be considered academic dishonesty for these project, which will lead to an automatic failure of this course.

If you use any resources that did not come directly from the class, you must specify and cite them in your code file. At the top of the starter code there will be a section to paste links to anywhere you got code or ideas from. This is to make sure that you are not plagiarising.

6 SQL Statements

All SQL keywords will be in ALL CAPS. For this project, the SQL keywords you need to handle are as follows:

CREATE TABLE
INSERT INTO VALUES
SELECT FROM ORDER BY

SQL data types (for this project) and their python equivalents are: SQL NULL = Python3 None
SQL INTEGER = Python3 int
SQL REAL = Python3 float
SQL TEXT = Python3 str

In our testcases we will use normal SQL syntax, and it will be up to you to convert that into python types. For example, in a CREATE statement we will refer to a column as type "TEXT" and you have to handle the conversion into Python.

7 SQL Syntax

7.1 CREATE TABLE

Example: CREATE TABLE stocks (symbol TEXT, price REAL, qty INTEGER);

This creates a table named "stocks" with three columns: column "symbol" composed to text strings, column "price" composed of real valued numbers (floating point), and column "qty" composed of integers.

7.2 INSERT INTO

Example: INSERT INTO stocks VALUES (’GOOG’, 40.4, 7);

Inserts the row (’GOOG", 40.4, 7) into the table named "stocks".

7.3 SELECT *

Example: SELECT * FROM stocks ORDER BY price;

Returns all the rows in the table named "stocks". The rows must be ordered by the column "price" ascending.

Given the above examples, the result should be a list of tuples: [(’GOOG’, 40.4, 7)]

7.4 SELECT COLUMNS

Example: SELECT qty, symbol FROM stocks ORDER BY price, qty;

Returns all rows, but only sohwing the qty and symbol values in that order. The rows are sorted by price (and where price is the same, by qty)

7.5 NULL values

Some tests will have NULL values. You need to represent this fact with the python object, None. When returning NULL v alues from a select statement, you should yield a python None object

8 Testing Your Code

8.1 Testing Yourself

We've provided a few sample sql transactions as well as a python script to test your code.

To run your code with the testing files provided:

```
python3 cli.py <filename>
```

Where the second argument is the .sql file to test. Each .sql file is a set of sql commands that compose one transaction. We also provide a way for you to compare against the ground truth, which is python's sqlite3 module. To run the ground truth, use the following command:

```
python3 cli.py <filename> --sqlite
```

Your output should be identical to the output from the sqlite3 module. In the cli.py file we have a few print statements so you know if you are using your own code or the ground truth sqlite3 module. You can remove this code if you like, in our final testing we will not have these print statements, of course.

You can, and should create your own test cases as well.

We also have a test file that you can run with hardcoded values that might make it easier to find errors. It is a normal python file that will return an error if your output is not correct. You can run this file as follows:

```
python3 test_project.py
```

There is no input file for this way of testing, because it is hardcoded.

8.2 Instructor Testing

When we test your code, we will run it in almost the exact same way, just with more test cases.

9 Assumptions and Guarantees

All tests will be legal SQL (no syntax errors, inserting into nonexistent tables or data type violations).
All select statements will have "FROM" and "ORDER BY" clauses
In "CREATE TABLE" statements every attribute will have a data type and no constraints
All table and columns names will start with a letter (or underscore) and be followed by 0 or more letters or numbers or underscores
No "TEXT" columns will contain single quotes

10 A Few Notes

The instructor solution for this project is 275 lines of code, including a lot of comments. This projet can be difficult, but if you understand how what is being asked the development itself should be reasonable.

11 Submission Instructions

You will submit a file named "project.py" to the handin (handin.cse.msu.edu). All of your code should be in the "project.py" file. If you used any external resources (something other than class material), make sure that you cite the resource in a comment at the top of your "project.py" submission file. The top of the starter code also has space to put your name, netid, PID, and an estimation for how long the project took you to complete. Make sure you fill this out fully before submitting your project.