Project 5: Database Programming with SQLite

Objectives
Understand how SQLite itself works from “the outside in” by exploring its API.
Learn how to write a simple C/C++ program that directly interfaces with SQLite.

Overview
Create a trivial clone of the `sqlite3` console program. It should allow your user to enter a simple SELECT statement and, in response, display the results of the query in a rough tabular format. Your program may then terminate, but you are encouraged to implement a simple read-eval-print loop allowing the user to execute multiple SELECT queries per session.

Consider the trivial code listing from class. Do you see the problem?

```
/* CSCI403 Project 5: Programming with SQLite
 * Author: Yong Joseph Bakos
 *
 * A simple program demonstrating interfacing with SQLite via its C api.
 * The program accepts a SQL query from the console and displays its results.
*/
#include <iostream>
#include "sqlite3.h"

static const char* dbPath = "albums.sqlite3.db";

static int tryToDisplayData(
    void* db,
    int numberOfColumns,
    char** values,
    char** columnNames) {
    for (int i = 0; i < numberOfColumns; ++i) {
        std::cout << values[i] << std::endl; // try hard
    }
    return 0;
}

int main(int argc, char* argv[]) {

    sqlite3* db;
    char* errMessage = NULL;
    int responseCode;

    responseCode = sqlite3_open(dbPath, &db);
    if (responseCode) {
        std::cout << "Error: " << sqlite3_errmsg(db);
        sqlite3_close(db);
        return 1;
    }
```

```
    std::cout << "Hello. I am SQLiiite.\n";
    std::cout << "> ";

    std::string query;
    getline(std::cin, query);

    responseCode = sqlite3_exec(db, query.c_str(), tryToDisplayData, 0, &errMessage);

    if (responseCode != SQLITE_OK) {
        std::cout << "Error: " << errMessage;
        sqlite3_close(db);
        return 1;
    }

    sqlite3_close(db);
    return 0;
}
```

Look at the callback function called `tryToDisplayData`.

Instructions
Read SQLite chapter 6 , The Core C API, pages 153 - 160. Note the section titled The Get Table Query and learn about `sqlite3_get_table`.

In your environment of choice, create a new C/C++ project.

Remember, you can use the linux virtual machine (link on Piazza Course Page), VS in the campus labs, or your own option. Want to try a fun IDE? Try QtCreator.

From this zip file add the files sqlite3.c, sqlite3.h and the database file, albums.sqlite3.db, to your project. Next, enter the simple main implementation above into your own source file.

Get this to compile and run. If you get stuck, ask on Piazza or drop by office hours!

Now, modify the code such that it uses the API function `sqlite3_get_table` to display the query results in a tabular format (a rendering of the relvar).

```
id | title | year | rank
1 | Love is a Foobar | 1999 | 1
...
100 | You Really Got Me | 1969 | 100
```

You may tidy the format, but make this your last priority.

Presentation
Submit your main code listing as an attachment on Blackboard.

Good writing is important, **especially when it comes to code**. *Writing with poor readability, lack of appropriate whitespace, misspellings, etc will be handed back for re-writing (no grade penalty).*

Submit a zip of your code via blackboard. I would prefer a zip containing only one source file (.c/cpp) rather than a bunch of project assets and the like.

Grading Criteria (60 points)
Complete implementation using `sqlite3_get_table` (60 total points).
- writing quality (30)
- implementation (30)

Please submit your work via Blackboard by 7AM on the due date.