CMPT291 - Introduction to File and Database Management

Lab 7

- Unless your lab instructor says otherwise, the due date for this lab is June 7th at 13:00. It is your responsibility to leave enough time to resolve unforeseen problems that may arise with the upload process.
- This lab is worth 4% of your final mark.
- Your submission must be a compressed archive (i.e. a TAR-GZ) file. No other format will be accepted. Your lab instructor will help you create this file.
- It is your responsibility to ensure your archive file extracts successfully on GitHub CodeSpaces, and that it includes the correct files and no additional files.
- The file names and formats must be exactly as described in this PDF.
- For questions that ask you to handwrite your answer, you must upload your legibly-written answer in JPG format **rotated to the correct orientation**. Each page must have your name on top and take up less than 300 KB of space. If you write your answer on paper, scan your page(s) via Microsoft Lens (works on both Apple and Android phones) or a similar scanning app.
- Your submission must be entirely your work. Your instructors reserve the right to ask you to explain your work in person at any time.
- Properly acknowledge (add a note and/or hyperlink and/or comment) any help or resource you used. This includes the software(s) you use to create pictures.
- It is your responsibility to periodically back up your files. Keep in mind that Gitub Codespaces are deleted every now and then.
- Failure to comply with any of the instructions will result in a mark of 0.

University Database

Student

sID	fName	lName	major
1	Alice	Johnson	Computer Science
2	Bob	Smith	Business
3	Carol	Lee	Computer Science
4	David	Kim	Mathematics
5	Eva	Brown	Psychology
6	Alice	White	Mathematics

Enrollment

sID	cID	grade
1	101	A
1	103	B
2	102	B
2	104	A
3	101	A
3	103	A
3	104	B
4	104	C
5	105	A
6	104	A
6	101	A

Course

cID	title	credits
101	Intro to Programming	4
102	Marketing Fundamentals	3
103	Data Structures	4
104	Statistics I	3
105	Intro to Psychology	3

Lab 7

In this lab, you use the `sqlite3` package in Python to create a program through which a user can interact with the university database. This package is comes with Python and there is nothing to download or install. That said, I recommend you do your programming on GitHub CodeSpaces. Refer to your CMPT200 notes if you need a refresher on Python.

1 Updating values in SQL

The syntax for updating a value in SQL is very simple. For example, recall the *Student* schema from the lectures. If you desire to update all the GPAs to 4.0, you can do it via the command

```
UPDATE Student SET GPA = 4.0;
```

But that destructive work is too generous to all. Suppose you only want to change the GPA of those who are named Marie. You do that like a selection:

```
UPDATE Student SET GPA = 4.0
WHERE fName = 'Marie';
```

Start SQLite3 in GitHub Codespaces and practice updating values.

What effect will updates have on `FOREIGN KEY` constraints? For example, if a student changes her `sID` will that new `sID` appear in the *Enrollment* table?

The answer is that just like every `FOREIGN KEY` declaration can have an `ON DELETE` clause, it can also have an `ON UPDATE` clause. The options for this clause are identical to that of `ON DELETE`.

Create a table with foreign keys in SQLite3 on Codespaces and attempt updating a referenced value. Once comfortable with updates, move on to the next section.

2 Exception handling in Python

Again, if you need a refresher, refer to your CMPT200 notes on this. In the particular case of SQLite, to prevent your program from crashing when SQLite returns an error, you can handle it like below:

```
conn = sqlite3.connect('example.db')
cursor = conn.cursor()

try:

cursor.execute("CREATE TABLE users (id INTEGER, name TEXT)")
cursor.execute("INSERT INTO users VALUES (1, 'John Doe')")
cursor.execute("INSERT INTO users VALUES (2, 'Jane Smith')")

## Commit transaction
conn.commit()
print("Transaction successful")

except sqlite3.Error as e:
## Rollback in case of error
conn.rollback()
print(f"Transaction failed: {e}")

finally:
#perhaps close the connection.
```

3 University database

Use the `sqlite3` module in Python to create and populate the University schema you see at the beginning of this lab. You must have appropriate constraints. If a student changes his/her foreign key value, the referencing table must show that updated value. If a student is deleted, the referencing tables should have no records for that student either. Your DB file must be named `uni.db`

When your program starts, the user must be greeted by the following screen:

```
Welcome to Cool University.

Enter the number of your choice.

"I would like to ..."
1. Insert a student
2. Enroll a student
3. Unenroll a student
4. Update a student record
5. See all the courses
6. See a student's enrollments
7. See how many are enrolled in a course
8. Exit
```

You can make use of the Python `input` function. Recall that this function returns strings of characters. Your string should be stripped of white spaces and checked for validity. If not valid print a message saying so, and re-show the screen. If valid, you must mention the input you interpreted. E.g.,

```
You entered 1. Insert a student
```

If 1 is indeed the chosen option you must one by one ask for the relevant attributes of the student. E.g.,

```
Please enter the student's ID:
```

Do not say `sID`. Those identifiers are internal to your program. Similarly, do not say `fName` – say first name instead.

Once you have shown/done any requested query/update (not just number 1), print the following after at least one empty line:

```
Press 1 to return to the main menu
Press 2 to exit
```

For choice 2, you must ask for the student's ID and the course's ID. Needless to say, if either the student or the course does not exist you must print an appropriate message.

Choice 3 is analogous.

Choice 4 should print the following:

```
Enter the student's ID whose record you want to update:
```

Once that ID is entered, either show an appropriate error message, or follow up by:

```
Which attribute of the student do you want to update?
1. The student's ID
2. The student's first name
3. The student's last name
4. The student's major
```

Choice 6 is analogous to choice 4.

Choice 5 simply prints all the courses.

Choice 7 asks for a course ID, and prints the course ID, title, and count of students in it.

Use the `tar` utility to archive and compress all your submission files. You must submit one tarball. In general, the name of your tarball should be `cmpt291_lab_N_X.tar.gz` where N is the lab number and X is your full initials. For example, for lab 1, Sam Porter Bridges would submit `cmpt291_lab_1_SPB.tar.gz` .

Submission: For this lab you must submit a tarball which includes:

1. `uni.db`
2. `uni.py`