Michelle Kutsanov

Rene German

May 18, 2021

CPSC 408

Final Write Up

My final project was a fun take on the Students Database we had worked with and modified earlier in the semester. I wanted to create a school featuring a new class of Marvel Superheroes (since I just finished all twenty three movies + three television shows and had the MCU on my mind from these past few weeks). The data I would include would be their first and last names, birthday, professors (Avengers members), RAs (non-Avengers cast members), and their Superpowers (varying from current known Marvel superpowers to ones I had imagined/always wanted).

A major problem that stems from a database with this vast amount of information is how to keep track of all this data without confusing or overwhelming the user. I am unfamiliar with how many other applications exist to query this data since I have never worked for a school district, but after my experiences working as the user with Blackboard, Canvas, and MyChapman, I have seen what I do not like in their UI and what troubled my parents when using them, so I made mine fairly bare bones. That was the main decision behind making this a console application.

The way I solved my aforementioned issue is that I utilized multiple View tables to make the user’s updates and searches very easily to manage. My program also has a linear-type of search where the user is able to keep making changes until an exit ‘e’ is entered into the input. I show my schema below in *Figure 1*- the user is asked at the end of every command “Would you like to keep manipulating the database (‘e’ to exit)?” Then the menu options are displayed for them.

If I break down the elements of my project, when the code is initially run, I offer the user a menu that allows them to perform one of many modifications/views of the data. If “1” is selected, they can view the schema as a whole. “2” is used for querying through the data through multiple parameters. “3” and “4” are selected if you would like to either add a record or delete a record- and this is done via the student’s StudentID. “6” is used to Update records currently in the database, and “5” is used to export all of the data in a specific table to a csv file named ‘marvel.csv’. Also, my usage of primary and foreign keys throughout the table helped enforce referential integrity across my database.

To break down the query option, if the user wanted to search through the professors or the superpowers, I utilized a join across all of the tables (MarvelStudent, StudentHomeroom, StudentHousing, and StudentPersonalInfo). If the user searched via birth year, the query contained a sub-query that asked for the oldest year that they wanted results filtered by. Finally, if the user wanted to get a count of how many students had each different Super Power, I utilized a Group By command to make this output more palatable. Figure 2 below shows the various ways I created and worked with these queries in MySqL.

What I struggled with most was probably how to Update the tables and having that update stick over all of the tables that reference each other but once I figured out how to do that through View tables, I resolved that issue. Another issue was burnout after trying to finish my last semester and feeling like I missed out on my senior year. Creating my data out of something I found joy in (the Marvel Cinematic Universe) helped me deal and come to terms with that problem.

Three things that are not ideal about my database are that (1) after the first initial run of the GenData() function, you have to comment it out since we do not want to overwhelm the application with too many faker data points. (2) Another issue I could not figure out was how to make ClassRoomNumbers correlate with Professors in the StudentHomeroom data table since ClassRoomNumbers utilized faker generated data from 1-9. This could be explained through Professors overseeing multiple classrooms (ClassRoomNumbers), just at different times because some schools are set up in this way. (3) Finally, when adding a student to the database, if the user enters the student's birthday in the wrong format, it is supposed to print a specific error message but instead the program just crashes- I sadly could not figure out how to fix that but I will just keep my fingers crossed that the user follows my instructions since I included the format needed in a print statement.

In terms of next steps, I think it would be interesting to make PersonnelID correlate with a photo of the student we are representing since I do not know how to work with that specific type of data type and it would be a fun challenge. Also, I would want to slowly implement Graphics, and would use PyQt to do so since that would simplify that process. Finally, someone in class used banners in their console app that I thought were very cool and would make the application more palatable for a user like my parents who prefer things that look cleaned up and “larger”.

In conclusion, I managed to create a fun, easy database to work with and I am proud of what I was able to create in my final semester here. I really enjoyed learning about SQL and thank you for a wonderful last semester Rene! I hope you have a great summer! (I hope you update the database to include Rene German, Superpower: Steak and Lobster!)

*Figure 1:* Schema Visualization of my project

*Figure 2:* Query Statements 