James Williams
CMSC 461
5/16/2018

Project Phase 5

Intro: What is in this Submission

First, let me mention everything that I am submitting. I am including the files that were submitted in all previous phases 1-4. The only phase that had any changes made to it was phase 4's sql scripts that generate the database. All files in the other three phases 1-3 are the exact same as when they were submitted from their respective due dates.

In phase 4, the only sql scripts that were modified were the table creation script, and the index creation and deletion script. The other scripts are left untouched. Only a few minor things had to be changed in the table creation script (nothing that would affect the schema). And in the Index creation and Deletion scripts, the only additional index that was added was an index to the books_used table. The only reason for this is because a table needs an index before it can sub-queried. All other tables that were sub-queried already had indexes that were put on them from phase 4.

For good measure, the scripts for generating the database are in both the phase 4 folder, and in the phase 5 folder (in a folder called 'DB Scripts'). They are the same scripts; it is because in phase 5, all the scripts to generate the database are a requirement.

In the phase 5 folder, there is the previously mentioned 'DB Scripts' folder with all the database generating scripts. There are also a few more files. There are the modules I created for the database in their respective python files. There is a sql script called 'queries_reports.sql' which is the sql file of all the scripts to support the Queries and Reports section of phase 5. And there is also a requirements.txt file which contains all the pip installs I have on my working machine. I think the pip requirements are so large because I have Anaconda installed on my computer, but I am afraid to shorten it in case it breaks the file. I'm sure for the modules I have include the only libraries I use are the python mysql.connector, PrettyTable, and python's built in library datetime.

For the modules I created 4 different python files. The modules for the UI Requirements and for the main part of Phase 5 are 'student_module.py', 'customer_support_module.py', and 'administrator_module.py'. I chose to separate the different modules in different files to emulate different users downloading different applications to perform their use cases. The 3 modules are named to their respective user. The only thing important to note is that **the super admin also logs into his module with 'administrator_module.py'.**

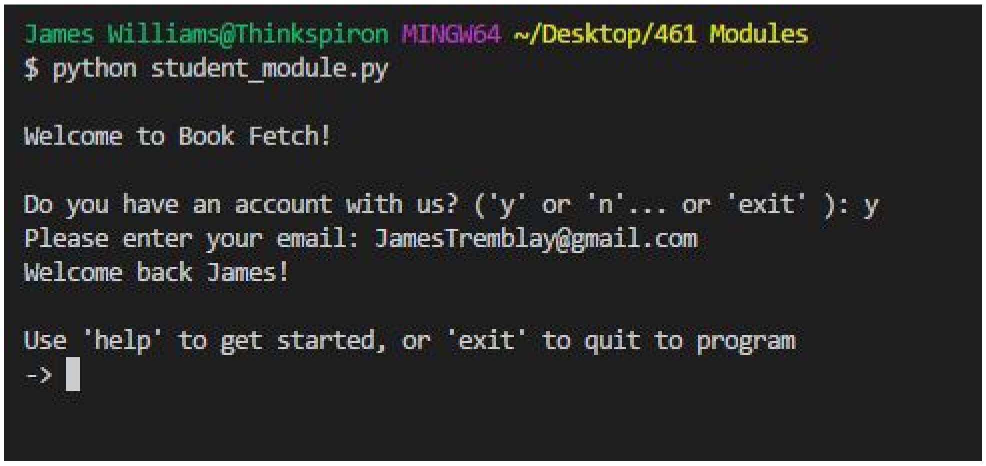
The fourth python file is called ‘queries_reports.py’. It is to satisfy the requirement that users can enter a number 1-24, and see the results of one of the queries made to fulfil the Queries and Reports section of phase 5. I chose to make it a separate module from the other three modules, because i didn’t want to put it in all three, nor did I want to choose just one to put it in. Anyone can open it up and enter a number to see the results of the query corresponding to the number in the the Queries and Reports section. I’m semi confident most of them are what was required for each query, **however I know for certain that queries 21 and 22 are not correct.** You can still call them and they will output something, but not the output that is expected.

Lastly, when using any of the four python modules, you will need to put your connection password in getConnection(), which is the first function of all 4 modules. Just find the string “YOUR_PASSWORD_HERE”, and replace it with the password to your connection.

User Modules

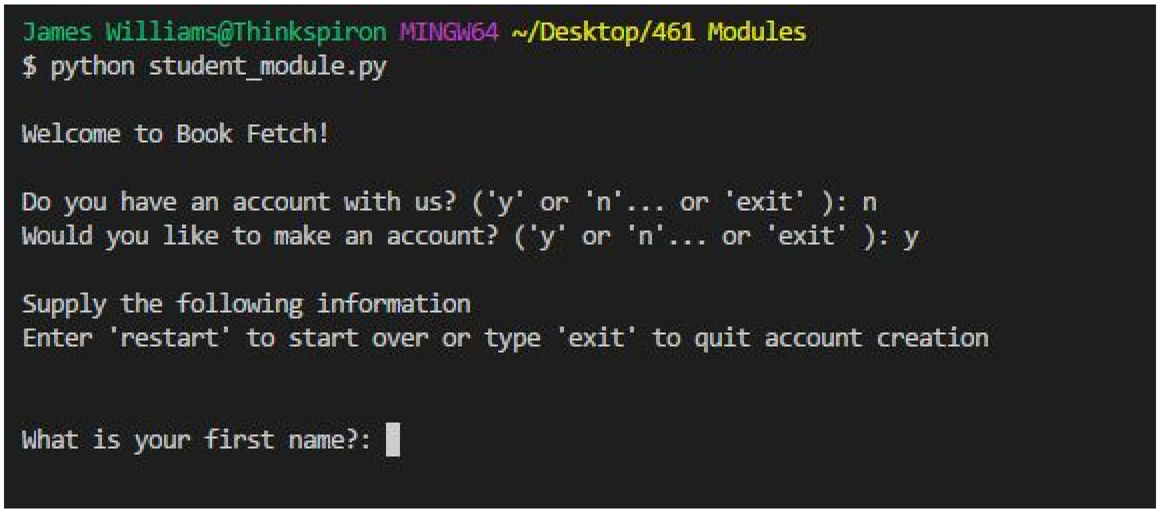
As mentioned previously there are 4 modules that I have submitted, ‘student_module.py’, ‘customer_support_module.py’, ‘administrator_module.py’, and ‘queries_reports.py’. The modules are named for their respective user, (exception being ‘queries_reports.py’ which anyone can use). I will go through each one briefly with the aid of screenshots

Student Module



```
James Williams@Thinkspiron MINGW64 ~/Desktop/461 Modules
$ python student_module.py

Welcome to Book Fetch!

Do you have an account with us? ('y' or 'n'... or 'exit' ): y
Please enter your email: JamesTremblay@gmail.com
Welcome back James!

Use 'help' to get started, or 'exit' to quit to program
->
```

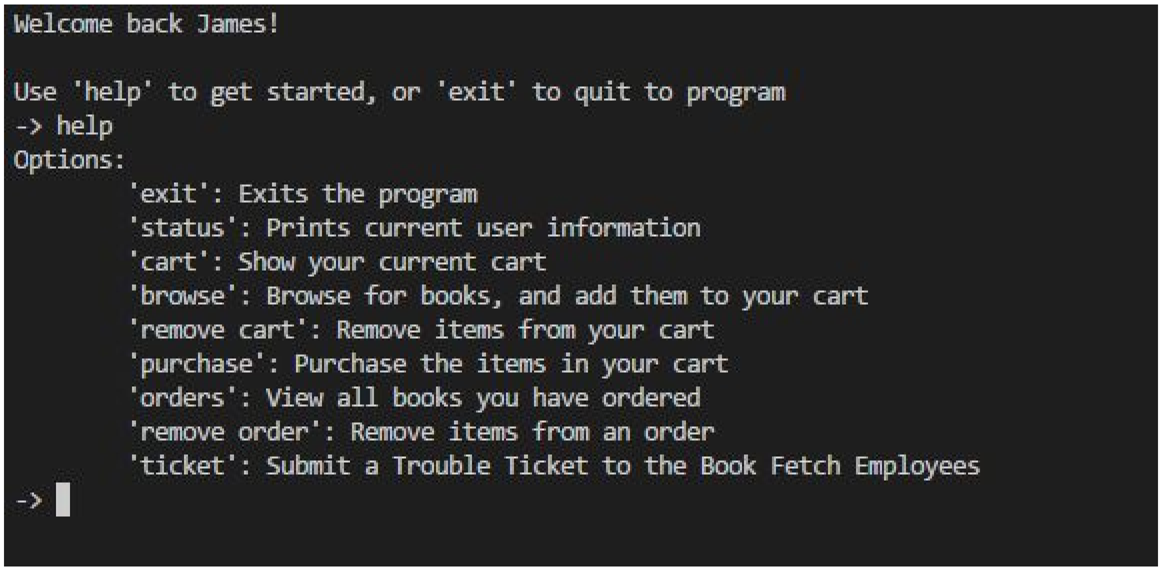
The student module can be opened in the terminal of your choice. Here I use bash on VS Code. Upon opening the student module you are asked to log in with just an email. If you do not have an email registered to the Book Fetch System, you can answer ‘n’ to the login, and answer ‘y’ when it asks you if you would like to create an account:



```
James Williams@Thinkspiron MINGW64 ~/Desktop/461 Modules
$ python student_module.py

Welcome to Book Fetch!

Do you have an account with us? ('y' or 'n'... or 'exit' ): n
Would you like to make an account? ('y' or 'n'... or 'exit' ): y

Supply the following information
Enter 'restart' to start over or type 'exit' to quit account creation


What is your first name?:
```

When navigating the menus, simply enter the information being requested by the module. When the module expects specific input, it will let you know by also print the options. **Note that input is not always verified however the program will never crash with invalid input.** If the program is asking for information, and the user supplies unexpected input, the query generated to interface with the database will raise an exception, but all exceptions are handled accordingly.

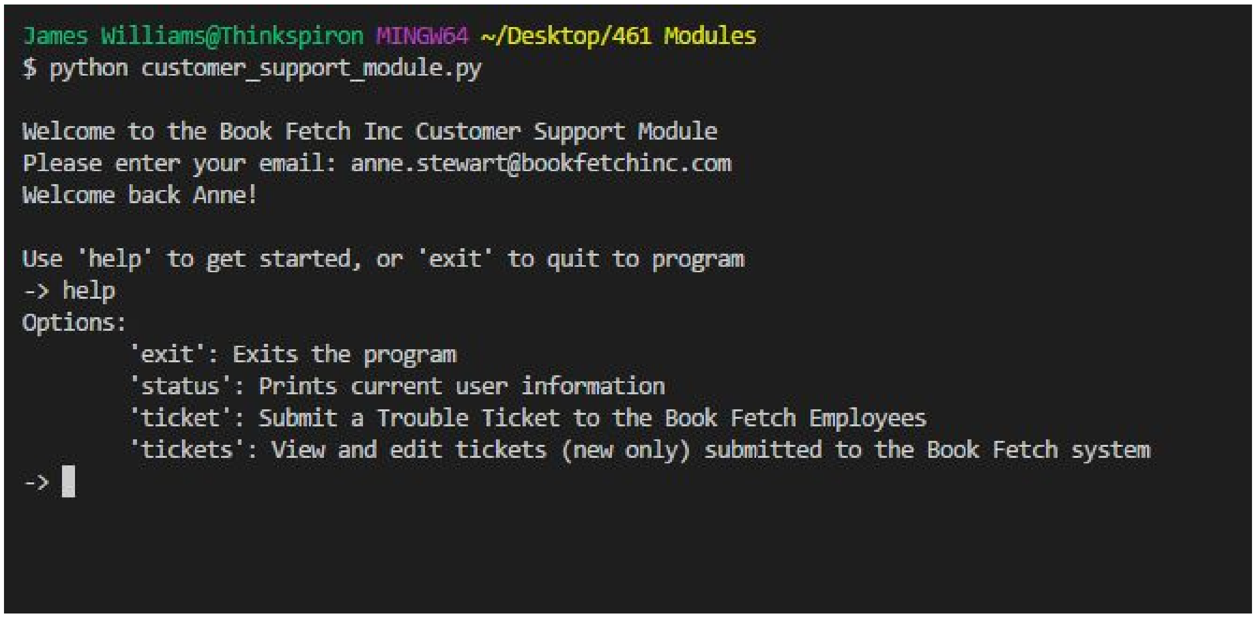
Upon logging into the a module, it acts similarly to a shell. The user has some option available to him. Upon entering 'help', the module prints all the options that are available to use:



```
Welcome back James!

Use 'help' to get started, or 'exit' to quit to program
-> help
Options:
        'exit': Exits the program
        'status': Prints current user information
        'cart': Show your current cart
        'browse': Browse for books, and add them to your cart
        'remove cart': Remove items from your cart
        'purchase': Purchase the items in your cart
        'orders': View all books you have ordered
        'remove order': Remove items from an order
        'ticket': Submit a Trouble Ticket to the Book Fetch Employees
->
```

The options listed in 'help' are directly made from the UI Requirements given in Phase 5. They represent the different use cases given to each user. Entering any one of these options will take the user to an appropriate menu, where they will be guided to enter the specified input. If a menu is looking for a specific string, then it will print the expected strings for the user to copy as his input. In many of the menus if multiple inputs are required in a row, the user will be given the option to enter ‘restart’ to start the input process over again if they make a mistake, or the option to enter ‘exit’ to quit the associated menu

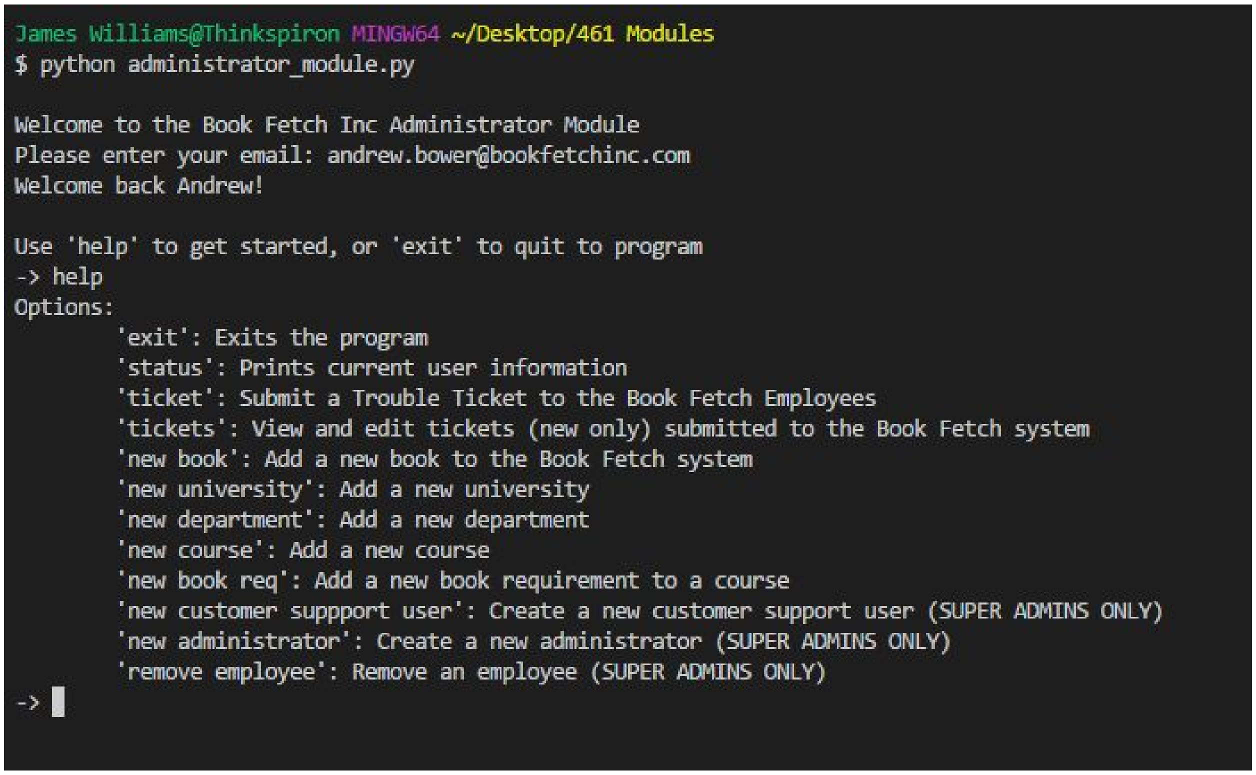
Customer Support Module



```
James Williams@Thinkspiron MINGW64 ~/Desktop/461 Modules
$ python customer_support_module.py

Welcome to the Book Fetch Inc Customer Support Module
Please enter your email: anne.stewart@bookfetchinc.com
Welcome back Anne!

Use 'help' to get started, or 'exit' to quit to program
-> help
Options:
        'exit': Exits the program
        'status': Prints current user information
        'ticket': Submit a Trouble Ticket to the Book Fetch Employees
        'tickets': View and edit tickets (new only) submitted to the Book Fetch system
->
```

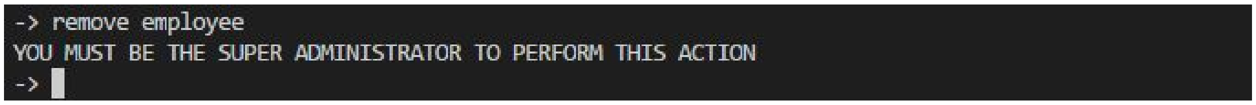
The employee modules work similar to the student module in terms of layout and navigation. This is the customer support module, and the usage is similar. Only this time when opening the module, the only option is login or exit. There is no option to make a new account because only the Super Admin can make new customer support users. Once again upon logging in, the user can use ‘help’ to get all the actions associated with them and these actions are based off the UI Requirements.

Administrator Module



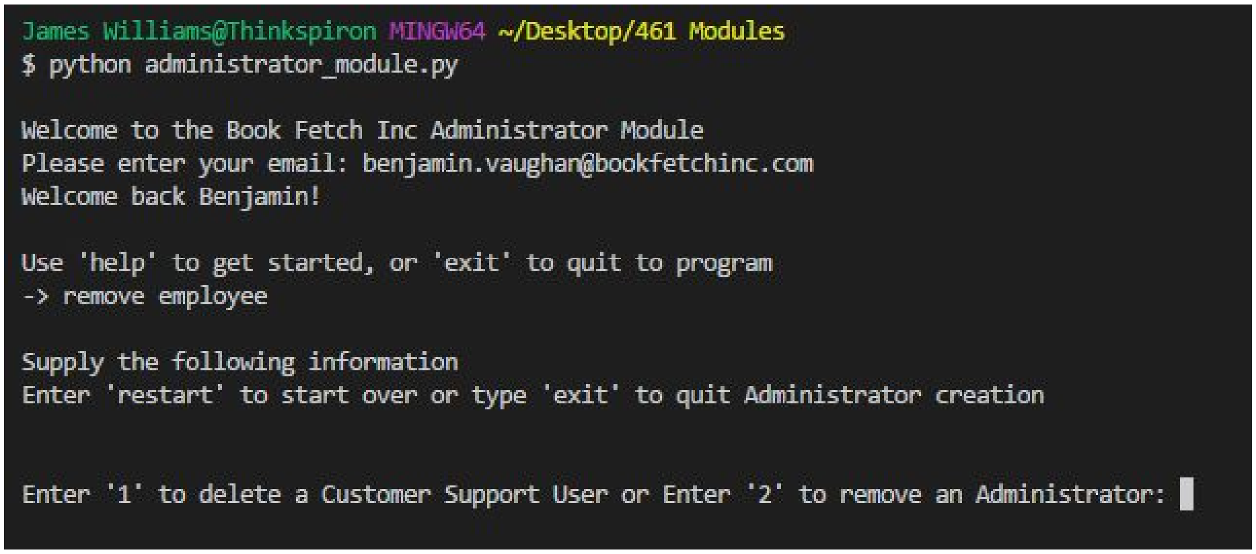
```
James Williams@Thinkspiron MINGW64 ~/Desktop/461 Modules
$ python administrator_module.py

Welcome to the Book Fetch Inc Administrator Module
Please enter your email: andrew.bower@bookfetchinc.com
Welcome back Andrew!

Use 'help' to get started, or 'exit' to quit to program
-> help
Options:
        'exit': Exits the program
        'status': Prints current user information
        'ticket': Submit a Trouble Ticket to the Book Fetch Employees
        'tickets': View and edit tickets (new only) submitted to the Book Fetch system
        'new book': Add a new book to the Book Fetch system
        'new university': Add a new university
        'new department': Add a new department
        'new course': Add a new course
        'new book req': Add a new book requirement to a course
        'new customer suppport user': Create a new customer support user (SUPER ADMINS ONLY)
        'new administrator': Create a new administrator (SUPER ADMINS ONLY)
        'remove employee': Remove an employee (SUPER ADMINS ONLY)
->
```

Once again, same thing as the customer support module, only this time their actions are for administrators. The important difference is that there are some options that require you to be logged in as the Super Administrator. These options are locked if you are not a Super Administrator:



```
-> remove employee
YOU MUST BE THE SUPER ADMINISTRATOR TO PERFORM THIS ACTION
->
```

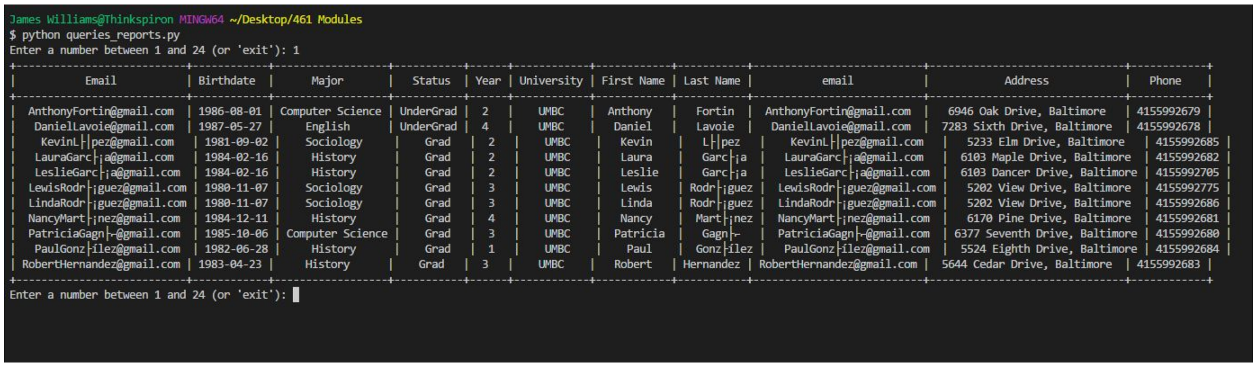
Here a regular Administrator is locked from doing these actions because he does have Super Administrator privileges. For the Super Administrator the session would look like this:



```
James Williams@Thinkspiron MINGW64 ~/Desktop/461 Modules
$ python administrator_module.py

Welcome to the Book Fetch Inc Administrator Module
Please enter your email: benjamin.vaughan@bookfetchinc.com
Welcome back Benjamin!

Use 'help' to get started, or 'exit' to quit to program
-> remove employee

Supply the following information
Enter 'restart' to start over or type 'exit' to quit Administrator creation


Enter '1' to delete a Customer Support User or Enter '2' to remove an Administrator: 
```

The one thing to note is the Super Administrator cannot delete himself. The only way for a Super Admin to be deleted is to be succeeded by another Super Administrator. To do so, the current Super Administrator must create a new Super Administrator, and have the new Super Administrator delete him. This is because there must always be at least one Super Administrator at a time, and this ensures the database is never left without a one.

Queries and Reports Module



```
James Williams@Thinkspiron MINGW64 ~/Desktop/461 Modules
$ python queries_reports.py
Enter a number between 1 and 24 (or 'exit'): 1
+---------------------------+------------+-------------------+-----------+------+------------+------------+-----------+---------------------------+--------------------------------+------------+
|           Email           | Birthdate  |       Major       |   Status  | Year | University | First Name | Last Name |           email           |            Address             |   Phone    |
+---------------------------+------------+-------------------+-----------+------+------------+------------+-----------+---------------------------+--------------------------------+------------+
|  AnthonyFortin@gmail.com  | 1986-08-01 | Computer Science  | UnderGrad |  2   |    UMBC    |  Anthony   |   Fortin  |  AnthonyFortin@gmail.com  |   6946 Oak Drive, Baltimore    | 4155992679 |
|   DanielLavoie@gmail.com  | 1987-05-27 |      English      | UnderGrad |  4   |    UMBC    |   Daniel   |   Lavoie  |   DanielLavoie@gmail.com  |  7283 Sixth Drive, Baltimore   | 4155992678 |
|    KevinL├│pez@gmail.com    | 1981-09-02 |     Sociology     |    Grad    |  2   |    UMBC    |   Kevin    |   L├│pez    |    KevinL├│pez@gmail.com    |    5233 Elm Drive, Baltimore     | 4155992685 |
|   LauraGarc├¡a@gmail.com    | 1984-02-16 |      History      |    Grad    |  2   |    UMBC    |   Laura    |   Garc├¡a   |   LauraGarc├¡a@gmail.com    |   6103 Maple Drive, Baltimore    | 4155992682 |
|   LeslieGarc├¡a@gmail.com   | 1984-02-16 |      History      |    Grad    |  2   |    UMBC    |   Leslie   |   Garc├¡a   |   LeslieGarc├¡a@gmail.com   |   6103 Dancer Drive, Baltimore   | 4155992705 |
|  LewisRodr├¡guez@gmail.com  | 1980-11-07 |     Sociology     |    Grad    |  3   |    UMBC    |   Lewis    | Rodr├¡guez  |  LewisRodr├¡guez@gmail.com  |    5202 View Drive, Baltimore    | 4155992775 |
|  LindaRodr├¡guez@gmail.com  | 1980-11-07 |     Sociology     |    Grad    |  3   |    UMBC    |   Linda    | Rodr├¡guez  |  LindaRodr├¡guez@gmail.com  |    5202 View Drive, Baltimore    | 4155992686 |
|  NancyMart├¡nez@gmail.com   | 1984-12-11 |      History      |    Grad    |  4   |    UMBC    |   Nancy    |  Mart├¡nez  |  NancyMart├¡nez@gmail.com   |    6170 Pine Drive, Baltimore    | 4155992681 |
|  PatriciaGagn├⌐@gmail.com   | 1985-10-06 |  Computer Science  |    Grad    |  3   |    UMBC    |  Patricia  |   Gagn├⌐    |  PatriciaGagn├⌐@gmail.com   |  6377 Seventh Drive, Baltimore   | 4155992680 |
|   PaulGonz├ílez@gmail.com   | 1982-06-28 |      History      |    Grad    |  1   |    UMBC    |    Paul    |  Gonz├ílez  |   PaulGonz├ílez@gmail.com   |  5524 Eighth Drive, Baltimore   | 4155992684 |
| RobertHernandez@gmail.com | 1983-04-23 |      History      |    Grad   |  3   |    UMBC    |   Robert   | Hernandez | RobertHernandez@gmail.com |  5644 Cedar Drive, Baltimore   | 4155992683 |
+---------------------------+------------+-------------------+-----------+------+------------+------------+-----------+---------------------------+--------------------------------+------------+
Enter a number between 1 and 24 (or 'exit'): 
```

The last module included is the module for viewing the results of all the queries assigned in the Queries and Reports section of phase 5. As mentioned previously, this module was given its own file, rather than being integrated into one of the three other user modules. This simple module only asks for a number between 1 and 24, and will execute the corresponding query on the database (see queries_reports.slq). The results of the queries are placed neatly (or as neatly as possible in a PrettyTable) and printed as shown here. I will admit this part was a bit rushed, and I cannot guarantee the queries work 100%. As I mentioned before I know that queries 21 and 22 are probably wrong, as I couldn't figure them out, so instead I gave my best 'non-crashing' versions of them.