Quickstart Guide:
CertRUs Database Utility

As part of the Trainee Database Construction Project
for CMSC 461-01, Sp 2020 at UMBC

Prepared by: Dylan Zach
Last Updated: 5/10/2020

Section 1: What is the CertRUs Database Utility?

The CertRUs database utility is a data storage system for the CertRUs company and its affiliates, which implements a MySQL database in order to solve the management of trainee certifications within the company. It consists of multiple parts, that being the User Interface coded in Python through Jupyter Notebooks, and the MySQL database that holds the data at hand.

In order to successfully use the contents of the CertRUs Database Utility, make sure that MySQL Server is installed on the machine and that it refers to port 3306 in its connections. Also make sure that a database schema called “certrus” is created, if it does not already exist. If you do not have root access, please ask your system administrator to generate a login and password for your account and relative privileges.

To open up the notebook, please install Anaconda, and more specifically Jupyter Notebook application from the appropriate source onto one’s machine. Once the steps are completed, open up the provided notebook file and run the code.

Section 2: Overview of the Login screen

Upon starting up the notebook and running, you will be prompted with a login interface. Simply input the username and password for your MySQL credentials in order to connect to the database.

```
=== Welcome to the CertRUs Database Utility! ===
Please enter your username
> not_root
Please enter your password
> ........
\\\Error: Username-password combination invalid!
Please enter your username
> root
Please enter your password
> ........
```

Section 3: How to interface with the database

Once you have established a connection with the MySQL server, you should be able to see an interface similar to the one pictured below. From here, you enter in the character corresponding in order to go from the submenu to the routine in order to perform the desired action. Each of these actions will be explained in-depth below.

Note that once an action completes, with the exception of quit, it will resume the loop indefinitely.

```
========================================================================
---- Welcome, root
---- What would you like to do?
'A': INSERT New data entries
'B': UPDATE Already existing entries
'C': DELETE From existing entries
'D': Generate a report of a dataset

'Q': To exit the utility
Type 'info' for information on the accessible data tables.
> info
You chose: info
The list of tables is:
certificates, challenge, challenge_event, challenge_skills, contact_info, co
q, score, skillmaster, skills, trainee
If you want to view a table's list of parameters, enter its name.
Otherwise, press ENTER to return.
>

========================================================================
---- Welcome, root
---- What would you like to do?
```

OPTION A/a: INSERT New data entries

In this section, you can create new queries for the database tables by inputting either “A” or “a”. It allows for insertion of a new tuple into any of the tables managed by the certrus database.

First, the program will prompt for choosing a table that will be inserted into. The list of tables can be viewed by running the “info” command at the starting list prompt. Note that the program is completely case-sensitive, and as such tables should be entered completely as-is, with no spaces or other special characters. Also, if one has already accessed a table during the lifetime of the program, then just entering ENTER will choose the last table accessed. This is helpful for if you need to input multiple entries into one table at one, and don’t want to re-type the table name for each iteration.

Secondly, assuming the table name is valid, you will be prompted to enter the values for the new query. You also get a response of the columns that are in the table that is requested. When entering a new query, be sure to enter the values in the EXACT order of the values that are returned as parameter names. Also, be sure that you do not include any commas except as dividers between entries, and do NOT PLACE SPACES AFTER THE COMMAS, unless you want the final data entry to have a space before its value in the database.

An example of a correctly formatted INSERT command is seen below:

```
Type 'info' for information on the accessible data tables.
> A
You chose: Option A/a
Which table should be inserted into?
(Press ENTER to modify last accessed table.)
> trainee
Parameter names of table "trainee", in order:
user_id,name
Enter the values for the new query, in order based on the parameters, separated by commas (,)
NOTE: Any space characters will be included as part of the entry!
> 2468,Hajime Hinata
SQL: INSERT INTO trainee (user_id,name) VALUES (2468, "Hajime Hinata")
Insertion successful. Return value:  None

========================================================================
```

Assuming that all values are inputted correctly, the “SQL:” line should be a valid SQL statement. Note that many of the tables rely on foreign keys in order to manage their database relations, so if you enter a value for a table that isn’t trainee, skills, certificates, or challenge, then you will likely run into an error if you don’t have the foreign key values on hand. For obtaining these when unknown, see “Option D/d” for report generation.

OPTION B/b: UPDATE Already existing entries

In this section, existing tuples that fulfill some criteria can be updated, or have specific values changed. This is helpful in cases such as where the name of a trainee needs to be changed in tables where they might appear, without modifying their user_id or existing entries.

The program prompts for multiple entries. On the first prompt, enter the exact table name that one wishes to access, much like with Option A/a. On the second prompt, enter in the conditional statements that one wishes to use to check to see whether or not a discovered tuple should be updated or not. Remember to enter the FULL conditional statement as-is. When separating with commas, be wary of spaces!

Next, the program prompts to enter which parameters will be changed in tuples where the conditional statements are all true. (In the current iteration, UPDATE only supports AND filtering for multiple statements.) In the final prompt, the program asks for what these parameters will be change to. Note that the order in which these are entered must be EXACTLY the order that the last prompt’s parameters were entered! This is important, as

otherwise data could be critically mismashed. And once more, be wary of white space! It's completely case-sensitive!

An example of a correctly formatted UPDATE statement is seen below:

```
> B
You chose: Option B/b.
Which table should be updated?
(Press ENTER to modify last accessed table.)
> contact_info
Parameter names of table "contact_info", in order:
user_id,street_adr,city,zip_code,phone_no,category
For which parameters checked should a statement be updated? Also include conditional statements.
Ex: "income > 50000"
(Delineate multiple by comma separation) (,)
> category = "None",zip_code = 0
Which parameters should be updated where last statement is true?
(Delineate multiple by comma separation) (,)
> category,zip_code
What should the values of the given parameter be changed to?
(Delineate multiple by comma separation) (,)
Note: Make sure that each values corresponds to the order of parameters in last statement!
> Default,90210
SQL: UPDATE contact_info SET category = "Default", zip_code = 90210 WHERE category = "None" AND zip_code = 0
Update successful. Return value:  None
```

For reference, here is the database after query "SELECT * FROM contact_info" before running the prior code:

user_id	street_adr	city	zip_code	phone_no	category
2468	124 Hopes Peak Blvd	Tokyo	0	555-0128	None

And after:

user_id	street_adr	city	zip_code	phone_no	category
2468	124 Hopes Peak Blvd	Tokyo	90210	555-0128	Default

OPTION C/c: DELETE From existing entries

Note in advance that depending on the level of permissions your account has, this statement may result in an exception statement no matter what. In this case, and for any other case where this happens, please contact your database administrator to obtain further permissions.

In this section, existing tuples that fulfill some criteria and are within a specified table can be deleted. Know that this option is very risky, and it is advised to be assured of your commands before executing this on the main database.

First, like the other options, the user is prompted to enter the table to delete entries from. Next, the user is prompted to enter a conditional statement on which to evaluate whether a tuple

should be deleted from the database or not. In most cases, a single one will do, but it is possible to evaluate for multiple. Again, you are responsible for your actions with this option!

A quick note that wasn't covered in other sections: for conditional statements, remember to enclose any string literals in quotations. This ensures that when passed to the database query handler, it returns the proper actions. Don't worry- it won't destroy the string. Python is good like that.

An example of using the delete command to remove a single entry:

```
========================================================================
---- Welcome, root
---- What would you like to do?
'A': INSERT New data entries
'B': UPDATE Already existing entries
'C': DELETE From existing entries
'D': Generate a report of a dataset

'Q': To exit the utility
Type 'info' for information on the accessible data tables.
> C
You chose: Option C/c
Which table should be deleted from?
(Press ENTER to modify last accessed table.)
> trainee
Parameter names of table "trainee", in order:
user_id,name
For which parameters checked should a statement be deleted? Also include conditional statements.
Ex: "income > 50000"
(Delineate multiple by comma separation) (,)
> name = "Sou Hiyori"
SQL: DELETE FROM trainee WHERE name = "Sou Hiyori"
Removal successful. Return value:  None

========================================================================
---- Welcome, root
```

user_id	name
1	Sou Hiyori
121	Mukuro Ikusaba
769	Reko Yabusame
2468	Izuru Kamukura
4727	Shin Tsukimi
9569	Mai Tsurugi
15056	Sara Chidouin
23035	Kazumi Mishima
23744	Shunsuke Hayasaka
36613	Keiji Shinogi
40068	Anzu Kinashi
44825	Nao Egokoro
48288	Kai Satou
54754	Naomichi Kurumada
55808	Kanna Kizuchi
56940	Alice Yabusame
60466	Megumi Sasahara
61189	Hinako Mishuku
65143	Gin Ibushi

user_id	name
121	Mukuro Ikusaba
769	Reko Yabusame
2468	Izuru Kamukura
4727	Shin Tsukimi
9569	Mai Tsurugi
15056	Sara Chidouin
23035	Kazumi Mishima
23744	Shunsuke Hayasaka
36613	Keiji Shinogi
40068	Anzu Kinashi
44825	Nao Egokoro
48288	Kai Satou
54754	Naomichi Kurumada
55808	Kanna Kizuchi
56940	Alice Yabusame
60466	Megumi Sasahara
61189	Hinako Mishuku
65143	Gin Ibushi

Left: Before. Right: After.

OPTION D/d: Generate a report of a dataset

This commands allows a user to make a SELECT query to the database, and then download that result as a .xlsx file. Again, make sure your account has proper permissions to make SELECT statements. (Though I don't see why it wouldn't, if you've been able to come this far…)

First, the program prompts for the name of the xlsx file that will be generated. Do not include .xlsx, as it will be appended by the program. Also, know that any symbols that can't be valid filenames will be excluded completely from the final file name.

Secondly, it prompts for the tables to include in the report. Unlike previous options, this one actually allows for inputting multiple tables, separated by commas. Again, it's case sensitive, so no spaces or capital letters. By default the tables will be joined sequentially using the NATURAL JOIN keyword. There is currently no support for other joins, but this might change, with feedback.

Thirdly, the program will prompt for which columns/parameters should be included in the final report. It also prints out a list of the parameters in each of the tables that were selected in the last step. They are separated by commas, as per usual. However! Make sure that IF an attribute shows up multiple times between the tables, that you specify WHICH parameter will be included using the dot operator (Ex: If user_id shows up in both trainee and has_cert, then specify either trainee.user_id or has_cert.user_id.) Pressing ENTER on this step will include all columns in both tables (wildcard).

Finally, you can enter conditional statements for query inclusion. This is whatever goes into the WHERE clause, if you're familiar with traditional SQL. You can also press ENTER to skip this step.

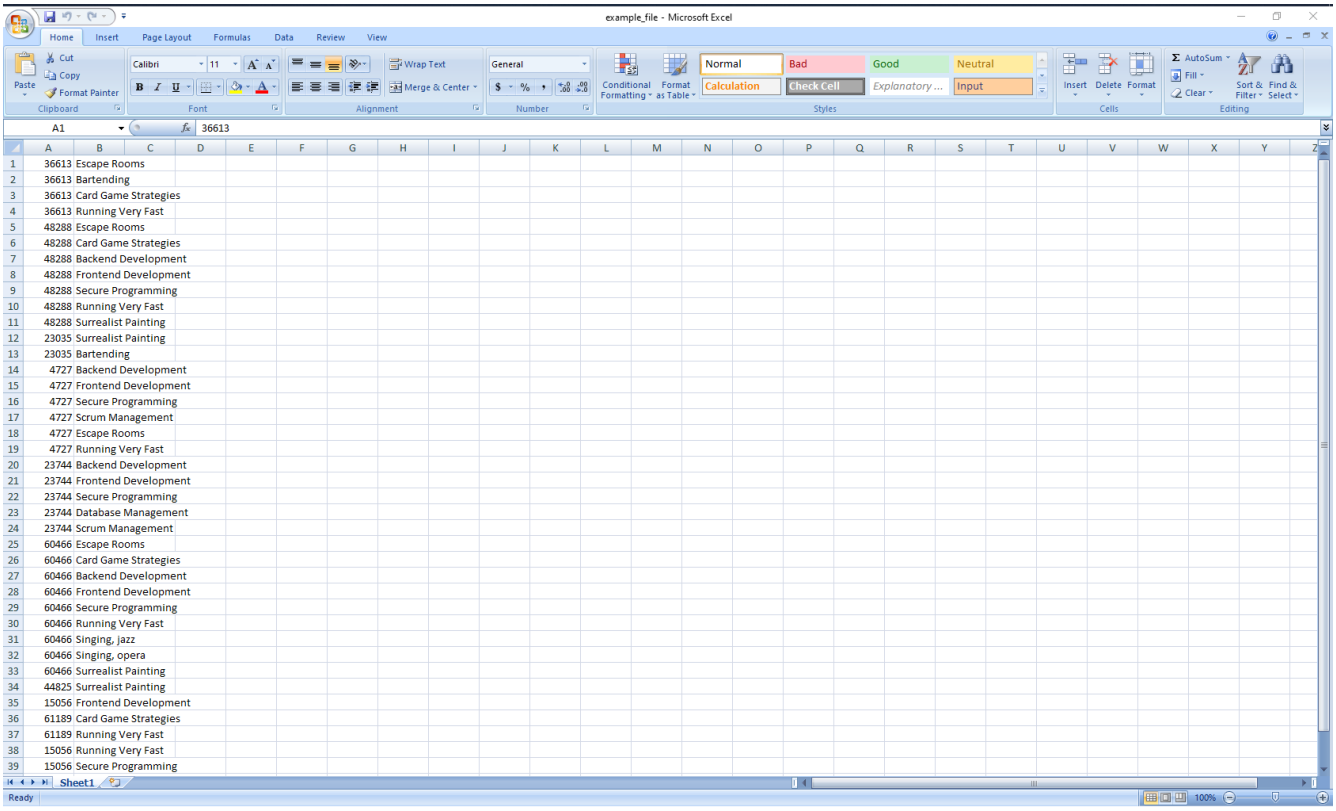
Should the SQL statement printed out be correct, your report will be generated into whatever folder your notebook is located inside of. It does not contain labels for the different columns, however – something that will be included in a future iteration.

An example of a properly-formatted report request can be seen below.

```
> D
You chose: Option D/d
What should the name of the report's output xlsx file be?
> example_file
Which tables should be included into the report?
(Delineate multiple by comma separation) (,)
They will be connected via natural join.
> trainee,has_skill
Parameter names of constituent tables, in order:
trainee: user_id,name
has_skill: user_id,skill_name,grade,challenge_inst_id
Which parameters should be included in report? (Press ENTER to include all)
(Delineate multiple by comma separation) (,)
For columns appearing in both sets, make sure to specify which by dot operator (Ex: account.balance)
> trainee.user_id,skill_name
Enter conditional statements for query inclusion, in parameter-value format.
Ex: "income > 50000"
(Delineate multiple by comma separation) (,)
(Press ENTER to include no special conditions)
> grade >= 3
Generating report... this may take awhile depending on sample space size.
SQL: SELECT DISTINCT trainee.user_id, skill_name FROM trainee NATURAL JOIN has_skill WHERE grade >= 3
SQL query processed, writing xlsx file...
Report successfully generated. Return value:  None

================================================================================
```

And, the resulting .xlsx file:



1	36613	Escape Rooms
2	36613	Bartending
3	36613	Card Game Strategies
4	36613	Running Very Fast
5	48288	Escape Rooms
6	48288	Card Game Strategies
7	48288	Backend Development
8	48288	Frontend Development
9	48288	Secure Programming
10	48288	Running Very Fast
11	48288	Surrealist Painting
12	23035	Surrealist Painting
13	23035	Bartending
14	4727	Backend Development
15	4727	Frontend Development
16	4727	Secure Programming
17	4727	Scrum Management
18	4727	Escape Rooms
19	4727	Running Very Fast
20	23744	Backend Development
21	23744	Frontend Development
22	23744	Secure Programming
23	23744	Database Management
24	23744	Scrum Management
25	60466	Escape Rooms
26	60466	Card Game Strategies
27	60466	Backend Development
28	60466	Frontend Development
29	60466	Secure Programming
30	60466	Running Very Fast
31	60466	Singing, jazz
32	60466	Singing, opera
33	60466	Surrealist Painting
34	44825	Surrealist Painting
35	15056	Frontend Development
36	61189	Card Game Strategies
37	61189	Running Very Fast
38	15056	Running Very Fast
39	15056	Secure Programming

OPTION info:

This option (by entering the full string “info”) shows which tables are actually present within the database. Entering in a valid table name will also show what attributes are present in this table.

Use this option if you want to view the attributes of a table, without actually making a query request.

OPTION Q/q:

Quits the program loop and closes the open connections. Select this when you’re done using the program!