Lab Assignment 4: SQLAlchemy Core

In this assignment we will start work on the sample project, by creating the database entries for it. You should download this Python script from GitHub[1] and this MySQL script from GitHub[2] and store them both in the same location where your keys.json file is. You will need to add a key called vault with value an object with keys username, password, server and schema. It would look something like this (we won't need the twitter part but you likely have it there already):

```
{
    "twitter": {
        "key": "....",
        "secret": "...."
    },
    "vault": {
        "username": "skiadas",
        "password": "....",
        "server": "vault.hanover.edu",
        "schema": "skiadas"
    }
}
```

You should use your own login for username and schema name, and type in your own password. Keep the server value at vault.hanover.edu as above.

You will be submitting two files. The one is an SQL script you should start. The other is the Python script you just downloaded, with your additions at the end. You should provide two solutions for each question:

- You should first work out the problem in the SQL script, working with MySQL-Workbench and your assignment4.sql file.
- Once you have that working, you should transport that solution into a SQLAlchemy solution back in the assignment4.py script.

NOTE: Both the Python script an the MySQL script start by dropping the previous tables, to make sure you have a clean start every time you run them.

The database will contain three tables:

- ev_users contains personal user information, such as a user's username, first and last name, and their affiliation or role.
- ev_events contains events. An event has an id, a title, some location information, start and end times/days, and an owner's username.
- ev_invites manages invites of users to events. Each row contains an event id, a user's username, and also an entry that represents the status of the "invitation".

Here are the questions. Remember that these questions must all be done BOTH in MySQL and in SQL Alchemy, unless the question says otherwise.

[1] https://github.com/skiadas/DataWranglingCourse/blob/gh-pages/assignments/assignment4.py
[2] https://github.com/skiadas/DataWranglingCourse/blob/gh-pages/assignments/assignment4.sql

1. The first step would be to write code that creates these three tables. First create the table ev_users. It should have the following columns/fields:

- username which is a variable length character string of length at most 20, it cannot be null and it is the primary key.
- first which is a variable length character string of length at most 40.
- last which is a variable length character string of length at most 40.
- affiliation which is a variable length character string of length at most 40, and should default to the string "None".
- In SQLAlchemy, store this table in a variable called tblUsers.

2. Next, create a table ev_events (with corresponding SQLAlchemy name tblEvents. It should have the following columns/fields:

- id which should be an auto-incrementing integer, not null and primary key.
- title which is a variable length character string of length at most 40, must be not null, and defaults to the empty string.
- longtitude which is an floating point number with 32 bits of precision.
- latitude which is an floating point number with 32 bits of precision.
- owner which is a variable length character string of length at most 20, it cannot be null, and it is a foreign key pointing to the username field of the ev_users table.
- start is a TIMESTAMP field in MySQL and a DateTime type is SQLAlchemy and must default to the value CURRENT_TIMESTAMP (which uses the current datetime when the entry is created) in MySQL and the value datetime.now() in SQLAlchemy.
- end is a TIMESTAMP field in MySQL and a DateTime type is SQLAlchemy and must default to null. You will have to enter NULL DEFAULT NULL after the TIMESTAMP part for MySQL to accept null as a valid timestamp value.

3. Next, create a table ev_invites (with corresponding SQLAlchemy name tblInvites). It should have the following columns/fields:

- event_id which is an integer, not null, and a foreign key pointing to the id entry in the ev_events table, with its "on delete" set to cascade.
- username which is a variable length character string of length at most 20, it cannot be null, and it is a foreign key pointing to the username entry of the ev_users table, with its "on delete" set to cascade.
- status should be an ENUM type, with possible values "Accepted", "Declined" and "Maybe". It should be allowed to be null. Read the MySQL documentation on enum types[3] and the SQLAlchemy documentation on the enum type[4] to find out how to do this. Make sure to understand how Python expects you to enter an enum value (it is not by simply providing a string, you have to create a class that represents the enumeration; The Status class has been created for you for this purpose).

[3] https://dev.mysql.com/doc/refman/8.0/en/enum.html
[4] https://docs.sqlalchemy.org/en/latest/core/type_basics.html

4. Write code that creates and adds a user named after yourself, with affiliation "Hanover College, Student", as well as another student user of your choice with the same affiliation, as well as a user named after a professor, with affiliation "Hanover College, Faculty, Staff".
5. Write code that creates a new event titled "Homecoming get-together", scheduled with a start timestamp of October 6th at 8am, with you as the owner, and with longtitude and latitude pinpointing to the Horner Center. A google search for Horner Center coordinates should return those values. By convention north and east values are positive, while south and west values are negative.
6. Write code that looks at any event where the owner of the event does not have an invite entry for the event itself, and inserts such an entry into the ev_events table with a status of "Accepted".
7. Write code that looks at any event whose title contains the word "Homecoming" and any user whose affiliation contains the phrase "Hanover College", and inserts invites to those users for those events (without a specified status). You will need to learn about SQL's LIKE string function[5] and the use of the % wildcard there.
8. Write code that looks at any event that has an end value of NULL or an end value earlier than the start value, and updates it so that it has an end value two hours after its start value. In order to do "time arithmetic", you will need to look up the details of the DATE_ADD[6] function in MySQL, and also the timedelta[7] object in the datetime module in Python (Python allows you to add a timedelta object to a datetime object).
9. (Tricky) Write code that looks at any event that has less than 5 people who have accepted their invite to it, and then reschedules it by moving the start date by one full day. You should start by creating a query that would return the ids of events that have less than 5 people accepted. You can do most of this using a GROUP BY together with the HAVING clause that you will need to read up on[8].

[5] https://dev.mysql.com/doc/refman/8.0/en/string-comparison-functions.html
[6] https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions.html#function_date-add
[7] https://docs.python.org/3/library/datetime.html#datetime.timedelta
[8] https://dev.mysql.com/doc/refman/8.0/en/select.html