Lab Assignment 5: Working with SQLAlchemy ORM

In this assignment we will do work similar to the previous assignment, but we will instead use the ORM system in SQLAlchemy where possible. Unlike assignment 4, in this assignment you will only work in Python. The relevant script is going to be this[1].

Recall that our database has 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".

We now need to think of this in terms of classes. You will create three classes, all inheriting from Base: User, Event, and Invite.

It is important that you add these classes in the designated section of the code, and not at the end. They must precede the metadata.drop_all(engine) and metadata.create_all(engine) lines.

1. Create a class User. It should have fields username, first, last, affiliation as in assignment 4.

2. Create a class Event. It should have fields id, title, longtitude, latitude, owner_name (in place of owner), start and end as described in assignment 4.

3. Add a bidirectional relationship between Events and their Owners. To do that you will need to add a relationship owner in the Event class, which points to the User class, and a relationship eventsOwned in the User class, which points to the Event class, and to include back_populates=... entries to both to indicate how they relate to each other.

4. Create a class Invite. It should have fields event_id, username, status as in assignment 4.

5. Create bidirectional relationships between User and Invite (calling the two ends invites and user respectively) and Event and Invite (calling the two ends invites and event respectively). Don't forget to include back_populates=... entries to link the two together.

6. Create User objects representing yourself and one professor as described in part 4 of assignment 4. Then add the two objects to the session.

[1]https://github.com/skiadas/DataWranglingCourse/blob/gh-pages/assignments/assignment5.py

7. Use a list comprehension to create 100 User objects whose usernames are "studentx", first names are "Numberx", last name is "Student", where x should be replaced by the incrementing numbers 1 up to 100. They all should have affiliation "Hanover College, Student". Use the session's add_all method to add all these to the session. Keep the list stored in some variable, as you will need to do more work to it.

8. Create an event "Homecoming get−together" as described in assignment 4 and add it to the session.

9. Use a list comprehension to create Invite objects to this event you created, for each of the 100 students you created. Add them to the session.

10. Use the random.choices function[2] of the random module to create a list of 100 status values chosen at random from the three options Status.Accepted, Status.Declined, Status.Maybe, assigning weights so that Accepted has a 30% chance of being chosen, Declined has a 30% chance also, and Maybe has the remaining 40% chance. The random package has been imported for you already.

    Then use a suitable loop to assign these to the corresponding invite objects you created in the previous part. You MUST use the zip function of python to create a list of pairs of an invite and a choice, then do a for loop over this list. so your answer should end up looking like:

    ```
    pairs = zip(..., ...)
    for invite, status in pairs:
        ... change the invite's␣status␣setting␣to␣provided␣value
    ```

11. You should be able to access the list of all invites to your event by doing something like ev.invites. Use a list comprehension starting from this list, to produce a list of strings, one for each invite with Accepted status, containing the first and last names of the invitee separated by a space. So your code might look something like:

    ```
    accepted = [
        ... something with invite.user ...
        for invite in homecoming.invites
        .....
    ]
    print(accepted)
    ```

[2] https://docs.python.org/3/library/random.html#random.choices