LAB-4b for Demo-App-Labs part 4 **(watch Video 04-DB as a lecture prereq for this lab)**
Lab Video: 04-database-try-sql

This lab should solidify your understanding of using Flask, flask_sqlalchemy, and python DB connections
This lab is more like a lecture--but you should follow along. I will answer questions in class and review.
After running the short examples from the lab-connect-db (lab 4a)
we try to do the same thing from within the sample Demo app that we examined.

Before continuing, make sure you have installed all database packages in your Anaconda environment

After you have watched the lecture video, let's apply it to our own database.
Use the video: 04-database-try-sql
Project folder: 04-Database-try-sql It's in the Demo-App-Lab folder
NOTE: This is from before we packaged as a project correctly.
flaskDemo.py
loops through, returns the entire string. Displays, but no template; uses fetchOne, not fetchAll
flaskDemo-template.py
loops through, passes string to template,
but renders the embedded html as literal characters!
when using a template, Flask does some preprocessing. Look at the source
flaskDemo-fetchall-template.py note (dictionary=True) access by field name!
pass the template the return from fetchall(), pass to template, iterate in tempalte!
works well!
now try to display just row.courseID. It doesn't recognize the field name from the table!
It will only recognize it if you use a flask model.
HOWEVER, if you try to display row[0], it will display the courseID. It's still a dictionary!
If you want to display row['courseID'], you have to return the query as a dictionary
dictionary = True in the connection string.

prereq video: 05-Packages (short lecture on package structure)
Video: 05-packages-try-ORM-model-reflect

Now, let's try to connect to a local/remote MySQL db using SQLAlchemy

```
mysql+mysqlconnector://<user>:<password>@<host>[:<port>]/<dbname>
```

We can define the model ourselves (using the ORM models):

05_Packages-try-ORM-models in Demo-App-Lab

Let's try some insert:

briefly show the Simple-CRUD-Flask or Simple-CRUD-Flask-Dept

This is using regular SQL Insert clauses. We want to use the ORM for this,

as it is more secure, and also seems more straightforward (no prepared queries, etc.)

Starting with 05-Packages-try-ORM-reflect

look at init, models and routes

We didn't define the tables as models in our models.py

Instead, see __init__: we imported models.py, and created the tables

See models.py: we used reflect!!

But the queries work!

Tried simple query in @home

Tried join queries (1:n and m:n) in @join

Clearly, it recognizes the table names and relationships.

(There are other ways of doing this, including explicitly defining the joins in the Model definitions.

But here, we don't have to pre-define anything.

We explicitly specify the join condition in the query.

Now, let's try some CRUD with your local database, Company

Videos:
08-Crud-Choices
08-Crud-Update-Delete-New

Start with 08-CrUD-Lab, (later, used as a starting point for CrUD-Lab-Choices)

__init__.py

update uri note that you will use localhost this time!!!
import models
models.db.create_all()
OF course, we have to update models.py to reflect the company db

models.py

db.Model.metadata.reflect(db.engine)
add the reflected models for db tables
you can leave the User and Post models in this application, to support the logins
The first time you run the application, you do not need "extend existing". Add after.

routes.py

import all of the models
Before trying anything more, let's test to see if our db is recongnized and imported properly:
in the def home, run a query similar to Select * from Department
a.k.a. Result = Department.query.all()
render dept_home.html

dept_home.html

in that template, keep href = #, because we're not ready yet to make indv dept page
display some of the fields, just as a test

Hopefully, this connected to the db and displayed some of the fields.
Now, add the href to the template, similar to the one used for post:

```
<h2><a class="article-title" href="{{ url_for('dept', dnumber=row.dnumber) }}">{{ row.dname }}</a></h2>
```

Note, I'm using "row" to iterate through the results; I'm using dnumber for the dept id.
Now that we have a link to a specific dept, where we can update/delete, we need to create those routes/templates

layout.html

14, 16, 23, 35, (66)

UPDATE and DELETE (INSERT too, but inserting a new dept doesn't use the "single dept" page)

Game plan and housekeeping

We will need to create new routes for viewing/updating/deleting a single dept.
These will be modeled off of the four "post"-related routes.
There are currently four "post" routes: post; post/new; post/update; and post/delete
So we need to create those four routes. (dept; dept/new, etc.)
Also, let's show the current date on some of the page.
The date of the user's post had been shown. But we don't have a post anymore.
So we'll use the current date, and pass it as a parameter to the templates.
from datetime import datetime
In app route /dept, we'll add the datetime as a parameter to send to the template
In the template dept.html, display the time as a method of the "now" datetime object that you passed.
Of course, since we're not storing the time, you could just use html and/or javascript to display the time.

To create a page where only one department is viewed. From this page, we can decide to update/delete.

@app.route("dept/<dnumber>") changed from def post
def dept(dnumber):
department = Department.query.get_or_404(dnumber)
return render_template('department.html', title=department.dname, dept=dept,now=datetime.utcnow())

dept.html

get rid of the if statement to test for current user. Let's assume for now that any user can make changes.
We can add something in later to incorporate different levels of users.
Similarly, change the image to default.html. The original image referenced the post.author.image.file
We don't have a post at this point, and so it is undefined.
Note that we kept the update and delete button (and that they are not the same--delete is danger!!!)
And we had to update the urls for those buttons.

To Delete--it's easier to understand and implement than update.

Changes made:
name of route, def, param
note use of "dept" and Department, just as before. I did this so thatyou can see the difference.
also redirected back to dept_home

Note: db.session.delete(it takes a class as argument, that is, it takes the model
Note: I had to redirect to "home", not to "dept_home", because there is no function (def) of dept_home.
This speaks to good design. Have a route for each page!! (Do as I say, not as I have shown you here.)

be sure to cascade changes to the templates and nav(in layout.html)

Lab assignment: If a new faculty can be added, we'd also like to enter which courses he or she is qualified to teach.

To Add a new department

Clearly, this will require a form. Model after a new post.
forms.py
Create a new form. Model after PostFc Name id DeptForm.
Be sure to import this form in routes.py!!!
Import an IntegerField, because our key is not autonumber.
Create a validation function to check for a unique dept id. (Will the db handle this? Yes, but throws an error.)
Since this will require a query on the Department model, be sure to import that model in this file.
Import a DateField, because we need the manager's start date.
(Perhaps if we were recording transactions, we would want a datetimefield
You might want to look at WTF form http://wtforms.readthedocs.io/en/latest/fields.html
Example using Regexp:
https://stackoverflow.com/questions/44614191/specify-detailed-format-of-validation-in-flask-wtforms
(We may want to use a Regexp for the social security number, as an example.)
https://www.safaribooksonline.com/library/view/regular-expressions-cookbook/9781449327453/ch04s12.html
https://stackoverflow.com/questions/48776006/regex-to-match-ssn-in-python
create_dept.html modeled after create_post.html
each field in the fieldset must reflect a field in the Department model. Cut/copy/paste

If you enter a valid ssn, but one that is not a current manager's ssn, you will get a DB-generated integrity error.
You can handle this with another def inside of the form's validation.
OR: you can limit the manager's ssns to the ones that are already in the db. (see below)
Similarly, the dname is specified in the company DB as being unique.
If you don't want a db-generated error, then you should def a validation (see choices code)

https://github.com/wtforms/wtforms/issues/2 This one worked!!!
see 08-CrUD-Lab-Choices

In *forms.py:*
See all of the imports for form fields and for form validators. Everything is an object/class and so imported.
Review the different fields for the form.
Tried to use Regexp for ssn. It works, but why make user enter the data?
Using a Regex for the date is a good idea!
Still wish we had an input template.
https://stackoverflow.com/questions/26057710/datepickerwidget-with-flask-flask-admin-and-wtforms
(import html5 widgets, e.g. datepicker)
Oh, that's nice. Notice that the pattern is mm/dd/yyyy,
even though we specified that it is stored as yyyy-mm-dd (check DB!)

from wtforms import SelectField
from wtforms.ext.sqlalchemy.fields import QuerySelectField
do not need this unless you are using that component, which I no longer am doing.
We created a query (ssns) that captures the "choices" for the SelectField.
The choices are a list of tuples (select-value, display-value)
The query actually returns only tuples. You could access it by row[0].
But if you want to access it by filed name, such as row['mgr_ssn'], change to dict
You could actually try to specify the choices inside the SelectField assignment, but w/a query, it gets hairy.
Note also that I am using the mgr_ssn as both the select field and the display-field.

To Update an existing dept: (model after update post)
Problem: In the User/Post example, the assumption was that the ID is an autonumber, and therefore
was never on the form for the user to input. Therefore, New Post and Update Post could
share the same form and the same validation logic.

However, in our case, our dnumber is not an autonumber. We do allow it to be entered on the
New Department form (DeptForm). When you try to update, you get an entity integriy error
due to the duplicate key.
Possible Solutions:
* determine how to conditionally display a field, or how to conditionally validate
This would require sending additional parameters to the form, and testing.
Not only can this get messy, possibly using the html template to test this (not good separation)
but it's complicated, probably also using the RequiredIf validator
https://gist.github.com/devxoul/7638142
https://www.codesd.com/item/how-to-make-a-conditionally-optional-field-in-wtforms.html
* You can create two separate forms. Possibly not requiring two separate templates (maybe yes)
Not a bad solution, if the two really require substantively different inputs

* You can create an update form, and then have the new form inherit from the update form.
  The new form can extend the update form with the dnumber input field and validation.
  https://stackoverflow.com/questions/35609669/conditional-wtfform-fields
  This is also a very good option, with less updating if there are structural changes to the DB.

→ We will attempt the last two options, in two separate projects

08-CrUD-Lab-Update creating a duplicate form, but using the same template for displaying

In *forms.py* , add a form DeptUpdateForm, removing the dnumber field and its validation.
In *routes.py,* import DeptUpdateForm, and fix the update dept route so it doesn't pass the dnumber either direction.
Can we use the same template, *create_dept.html* ?
  Test to see if there's a form.dnumber--enclose that entire div in an if statement
  Alternatively, create a separate template called update_dept.html

08-CrUD-Lab-Inherit New form inherits from Update form, adds field for ID (dnumber, in our case) and def for ID validation

In *forms.py* , change DeptForm to inherit from DeptUpdateForm. Be sure to put DeptForm after DeptUpdateForm.

look at Inherit-Problem dept name still can't be the same upon update. (see Forms and create.dept.html)
Look at Inherit-Solution, which works, but leaves a hidden field of dnumber, and a big space on the output.
Look at inherit-template--now we got rid of the problem.

https://stackoverflow.com/questions/29169699/flask-wtforms-populating-drop-down-list
https://stackoverflow.com/questions/43548561/populate-a-wtforms-selectfield-with-an-sql-query

IF you look at the link below, you can see how the create_dept.html uses a Flask macro to replace a regular form html
https://www.youtube.com/watch?v=eu0tg4vgFr4 (slightly customized)

http://wtforms.readthedocs.io/en/latest/fields.html#html5-fields
http://flask.pocoo.org/snippets/60/

https://stackoverflow.com/questions/17887519/how-to-use-queryselectfield-in-flask

http://regexlib.com/DisplayPatterns.aspx?cattabindex=4&categoryid=5&p=7

https://stackoverflow.com/questions/31624530/return-sqlalchemy-results-as-dicts-instead-of-lists