4820 - Assignment 02 - Parse text and load to database

Pts: 200

Due May 7th

The most common method for custom data to be loaded into the database is to read a file of data, convert some of that data into insert statements and produce as output a .sql file. Sometimes this is taking data from an existing database (via a select statement) and then using that as the input. The end result is a set of “insert” statements.

In this assignment use Python to write a program that will read in the set of `*.md` files in the class githbub directory learn-db/hw, https://github.com/Univ-Wyo-Education/S21-4820/tree/master/learn-db Parse each file. Look for `# Interactive` on a line to find the title of the interactive homework. Look for `#### Tags` to get a comma separated list of tags. Each line has a series of values.

You are inserting into these tables:

```
CREATE EXTENSION if not exists "uuid-ossp";
CREATE EXTENSION if not exists pgcrypto;

DROP TABLE if exists a02_title_to_tag cascade ;
DROP TABLE if exists a02_title cascade ;
DROP TABLE if exists a02_tags cascade ;

CREATE TABLE a02_title (
        id uuid DEFAULT uuid_generate_v4() not null primary key,
        title text not null,
        body text not null
);

CREATE TABLE a02_title_to_tag (
        id uuid DEFAULT uuid_generate_v4() not null primary key,
        title_id uuid not null,
        tag_id uuid not null
);

CREATE UNIQUE INDEX a02_title_to_tag_u1 on a02_title_to_tag ( title_id, tag_id );
CREATE UNIQUE INDEX a02_title_to_tag_u2 on a02_title_to_tag ( tag_id, title_id );

CREATE TABLE a02_tags (
        id uuid DEFAULT uuid_generate_v4() not null primary key,
```

```
        tag text
);

CREATE UNIQUE INDEX a02_tags_p1 on a02_tags ( tag );


ALTER TABLE a02_title_to_tag
        ADD CONSTRAINT a02_title_id_fk1
        FOREIGN KEY (title_id)
        REFERENCES a02_title (id)
;
ALTER TABLE a02_title_to_tag
        ADD CONSTRAINT a02_tag_id_fk2
        FOREIGN KEY (tag_id)
        REFERENCES a02_tags (id)
;
```

Create the tables in your system. Write a script that will delete all the data from the tables so when your insert's don't work you can cleanup and start over. (Save the delete script and turn it in also).

Your Python program should take a list of files to parse.

So…

```
$ python assignment02.py hw03.md hw04.md >inserts.sql
```

Will output to stdout the results for parsing hw03.md and hw04.md. The in the example the output is piped to inserts.sql.

Then you should be able to

```
$ psql
pschlump=# \i cleanup.sql
pschlump=# \i inserts.sql
...
psqhlump=# \q
```

and load the data.

Outline

1. First get a copy of the ‘git’ repository. Navigate to the top level of the github.com for this class. Then do click on the green button that says *Code* on the right. A pop-up appears. It will have 1 or more link that you can copy that end in `.git`. Probably pick the 1st one, that is using https to clone the repository. In a appropriate work directory, at the command line

```
$ git clone https://github.com/Univ-Wyo-Education/S21-4820.git
```

2. This will get you a current copy of the entire repository. Now navigate to the learn-db/hw directory inside the repository. Use `ls` to get a list of the input files. You want to process all the `*.md` files and skip all the `*.raw.md` files.

```
$ cd S21-4820/learn-db/hw
$ ls *.raw.*
$ ls *.md
```

3. Write your python program to access the command line. As a temporary test echo out file names.
4. Open each file. As a temporary test just copy it out. This will verify that you are reading the data.
5. Read each file one-line at a time - or take the text of each file and split it into lines.
6. Find the titles and body.
7. Create insert statements for the title and body. You will need to generate the UUID for each in the python code. An important note is to look for single quotes `` in the data. Replace the single quotes with 2 occurancs, “` to make the insert work.
8. Find the `#### Tags` line - parse it into a list of tags. Python has a good CSV parsing library. Use that.
9. Build a dictionary of “tags”. For each entry in the dictionary record the UUID of the document as an element in a list.
10. Walk across the keys to the dictionary outputting insert statements for each tag into the `a02_tags` table. You will need to keep a list or dictionary of UUIDs and associated tags as you are doing this.
11. Generate the insert statements for the `a02_title_to_tag` table. This table associates documents to tags in a M:N relationship.

```
$ psql
pschlump=# select count(1) from a02_tags;
count
```

```
-------
107
(1 row)

pschlump=# select count(1) from a02_title_to_tag;
count
-------
217
(1 row)
```

12. Save this code - we will be using it again in Assignment 03 as our source of data.

Turn In

1. Your python program
2. Your cleanup script
3. Your insert statments - generated by (1) abvove.