Assignment 9 – database

due in class on Mon 28 Apr (40 points)

In this assignment, you will learn a bit about the Structured Query Language for databases (SQL), and use `SELECT` statements to answer questions about the records in our movie database.

In response to each question, you should provide the requested data **as well as** the query you typed to produce that data. (I can show you how to copy and paste out of your terminal connection.)

You may do this in groups of three, but as in the cloud assignment, please rotate responsibilities of reading, typing, and taking notes.

Connecting to the database

On Windows, open PuTTY and use the host name `imdb.liucs.net`, user name `www-data`, and a password I'll provide in class.

On the Mac, open the Terminal application, and type:

```
ssh www-data@imdb.liucs.net
```

Using either platform, once you see the server's prompt:

```
www-data@testbed:~$
```

You should type:

```
./dbshell
```

The next prompt you see will be:

```
db101=>
```

This means you're talking to the PostgreSQL (psql) database on the Rackspace server in Virginia that contains our miniature IMDB clone.

A note about the prompt: When the database prompt ends in `=>`, that means it is ready for a new SQL command. When it ends in `->`, it is expecting you to continue the previous command. Often this means that you forgot to type the semicolon `;` at the end of your command. If you forgot, it's okay to type it now and press enter.

The tables

The first thing we'll do is ask PostgreSQL what tables it has in the database. Type the command backslash and lowercase d: `\d` and press enter. You'll see a list of tables that includes `django_admin_log`, `movies_movie`, and `users_user`. At this point, psql may put you in "pager mode," with the prompt `:` or `(END)`. When in pager mode you can type space to move forward, b to move back, or q to exit the pager back to your regular psql prompt.

The tables we'll focus on are those having to do with movies. You can get a more detailed description of a table by typing

```
\d movies_person
```

You'll see a list of the fields and their data types. For your reference, the fields and types within each movie table are given below.

```
CREATE TABLE "movies_person" (
    "id" integer NOT NULL PRIMARY KEY,
    "first_name" varchar(64) NOT NULL,
    "last_name" varchar(64) NOT NULL,
    "birth_date" date,
    "imdb_key" varchar(64),
    "created_by_id" integer NOT NULL REFERENCES "users_user" ("id"),
    "created_at" datetime NOT NULL
);
CREATE TABLE "movies_movie" (
    "id" integer NOT NULL PRIMARY KEY,
    "title" varchar(128) NOT NULL,
    "year" integer NOT NULL,
    "director_id" integer REFERENCES "movies_person" ("id"),
    "created_by_id" integer NOT NULL REFERENCES "users_user" ("id"),
    "created_at" datetime NOT NULL,
    "imdb_key" varchar(64)
);
CREATE TABLE "movies_role" (
    "id" integer NOT NULL PRIMARY KEY,
    "movie_id" integer NOT NULL REFERENCES "movies_movie" ("id"),
    "actor_id" integer NOT NULL REFERENCES "movies_person" ("id"),
    "character" varchar(64) NOT NULL,
    "created_by_id" integer NOT NULL REFERENCES "users_user" ("id"),
    "created_at" datetime NOT NULL
);
```

The format of the table descriptions above is not exactly what you'll get from psql's `\d` command, although the field names and types will match. The code above consists of SQL statements that would allow us to recreate these tables on a new server.

Databases support four basic kinds of operations, sometimes known by the memorable acronym "CRUD."

Acronym	SQL verbs
Create	CREATE, INSERT
Retrieve	SELECT
Update	ALTER, UPDATE
Delete	DROP, DELETE

Apparently the designers of SQL thought the acronym was undignified, because they called their retrieve operator "Select." The rest of this assignment is about using SQL select statements.

Single-table queries

The basic form of a select query is "select [fields] from [table];" where you replace the parts in brackets. The simplest way is to just retrieve **all** the fields, which is written with an asterisk:

```
select * from movies_movie;
```

You should see a list of all the data about movies. It will probably enter pager mode, so remember that you can use space to move forward, or q to exit.

1. Try the SQL queries to show data from the person and role tables too. Provide the select statements you used in your answers.

   The output can be a bit messy if the table has many fields, especially if they don't all fit across the screen. A solution to that is to specify just those columns you want, like this:

   ```
   select title, year from movies_movie;
   ```

2. Try an SQL query to show first name, last name, and birth date in the person table. Provide the correct select statements in your answers.
3. Try an SQL query to show just the character name and movie ID in the role table.
4. Do the records you're seeing appear to be in any particular order?

Limits and ordering

```
select title, year from movies_movie order by year;

select title, year from movies_movie order by year desc;
```

The `desc` is for **descending** order.

```
select first_name, last_name, birth_date from movies_person
order by birth_date;
```

Limiting number of records:

```
select title, year from movies_movie order by year desc limit 1;
```

5. Use the techniques in this section to print the first name and last name of the first 10 people, as alphabetized by their first name. (Hint: Andy Wachowski seems to be the first one.)

Where clauses

Filtering records:

```
select title, year from movies_movie where year < 1985;

select first_name, last_name, birth_date from movies_person
where birth_date < '1945-01-01';

select first_name, last_name, birth_date from movies_person
where date_part('month', birth_date) = '03';

select first_name, last_name, birth_date from movies_person
where first_name ilike 'j%';
```

6. List all the movies released in 2013 whose titles begin with "The".

Joining tables

To query fields from different tables we need to join them together using foreign keys.

```
select title, first_name, last_name
from movies_movie m, movies_person p
where m.director_id=p.id;

select character, title, last_name
from movies_role r, movies_movie m, movies_person p
where r.movie_id = m.id and r.actor_id = p.id;
```

7. Show the titles of all the movies directed by someone whose first name begins with a "C".