DSCI 551 – Fall 2021

Homework 3 (SQL) Solution, 100 points

Due: 10/24, Sunday, 11:59pm

In this homework, install the Sakila database as described in https://dev.mysql.com/doc/sakila/en/.

Or you may follow these steps to install it on EC2.

- Download package:
  - wget https://downloads.mysql.com/docs/sakila-db.tar.gz
- Unzip it:
  - tar xvf sakila-db.tar.gz
- Install:
  - cd sakila-db
  - mysql -u root -p < sakila-schema.sql
  - mysql -u root -p < sakila-data.sql
- Now log in to mysql, you should see the *sakila* database.
- Run the following command in mysql, if you haven't created a user named “dsci551” with password “Dsci-551” in mysql, please refer to lab 1.

  ```
  GRANT ALL PRIVILEGES ON sakila.* TO 'dsci551'@'localhost';
  ```

- Download “hw3_grade.sh” from blackboard and put it in the directory (e.g LASTNAME_FIRSTNAME_HW3) you are working on
  a. cd LASTNAME_FIRSTNAME_HW3
  b. chmod 707 hw3_grade.sh

1. Please write SQL query for each of the following questions. (50 pts, 5 pts each)

   Submission format:

   For each problem, create a file named “q1_<problem_index>.sql”,

   For example, for problem a, “q1_a.sql”

   Inside your sql files, it should look like this use sakila;

<your sql query>

a. Find actors in the actor table whose first name contains “er”. Return all columns. Your columns’ names and order should look **EXACTLY** like

```
+----------+-------------+-----------+---------------------+
| actor_id | first_name  | last_name | last_update         |
+----------+-------------+-----------+---------------------+
```

use sakila;

select * from actor where first_name like "%er%";

b. Find the second highest amount in the payment table using order by and limit. Return the amount only. Your column name should look **EXACTLY** like

```
+--------+
| amount |
+--------+
```

use sakila;

select amount from payment order by amount desc limit 1 offset 1;

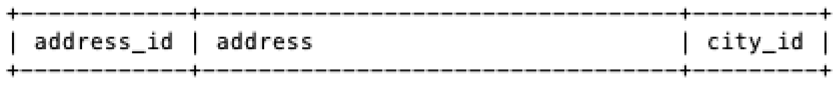
c. Find all films acted by the actor with actor_id = 1; Return actor_id, first_name, last_name, film_id, film_title. Your columns’ names and order should look **EXACTLY** like

```
+----------+-------------+--------------+----------+------+

| actor_id|first_name| last_name | film_id | title  |

+----------+-------------+--------------+----------+------+
```

use sakila;

select a.actor_id, first_name, last_name, film_id from actor a join film_actor fa on a.actor_id = fa.actor_id where a.actor_id = 1;

d. Find all store addresses that are in Argentina (country_id=6; you can use this information directly); Return address_id, address, and city_id. Your columns' names and order should look **EXACTLY** like

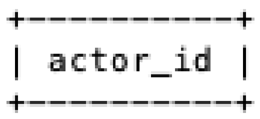


```
+------------+----------------------------------------+---------+
| address_id | address                                | city_id |
+------------+----------------------------------------+---------+
```

use sakila;

select address_id, address, a.city_id from address a join city c on a.city_id = c.city_id where c.city_id in (select city_id from city where country_id = 6);

e. Find all actors who have played in at least 1 film that is shorter than 48 minutes (length < 48); Return distinct actor_id only, in **ascending order**. Your column name should look **EXACTLY** like

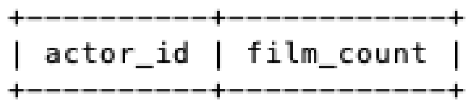


```
+----------+
| actor_id |
+----------+
```

use sakila;

select distinct actor_id from film_actor where exists (select * from film where film.film_id = film_actor.film_id and length < 48) order by actor_id asc;

f. Find the top 5 actors who have played in most films based on records in the film_actor table; Return actor_id and the count of films played (name this column film_count); sort the result by film_count in descending order. Your columns' names and order should look **EXACTLY** like



```
+----------+------------+
| actor_id | film_count |
+----------+------------+
```

use sakila;

select actor_id, count(film_id) as film_count from film_actor group by actor_id order by film_count desc limit 5;

g. Find the actors who acted in more than 30 films. Show actor names in ascending order by first name then last name. Your column names and order should look **EXACTLY** like:

```
+---------------+---------------+
| first_name  | last_name   |
+---------------+---------------+
```

```
use sakila;

select first_name, last_name
from film_actor as fa join actor as a on fa.actor_id = a.actor_id
group by a.actor_id
having count(distinct fa.film_id) >30
order by first_name, last_name;
```

h. Find the languages that are not presented in any films. Sort the result in ascending order.

```
+----------+
| name     |
+----------+
```

```
use sakila;

select name from language
where language_id not in (select distinct language_id from film)
order by name;
```

i. Find out how many different categories of films Ed Chase has appeared in. Your column names and order should look **EXACTLY** like:

```
+-----------------------------+
| number_of_categories |
+-----------------------------+
```

```
use sakila;
select count(distinct name) as number_of_categories
from category
join film_category
	on film_category.category_id = category.category_id
join film
	on film.film_id = film_category.film_id
join film_actor
	on film_actor.film_id = film.film_id
join actor
	on actor.actor_id = film_actor.actor_id
where first_name = 'ED' and last_name = 'CHASE';
```

j. Use Any to find the *title* and *release years* of all films that the actor_id =1 has acted in. Sort the result by title in ascending order. Your column names and order should look **EXACTLY** like:

```
+------------------+-------------------+
| title            | release_year  |
+------------------+-------------------+
```

```
use sakila;

select title, release_year from film as f

where film_id = any(select film_id from film_actor where actor_id = 1)

order by title;
```

2.Create a view table called 'Comedy_film' that contains all the films in the 'Comedy' category. You can design your own view table (select columns you need) to meet the requirements below. (25 pts)

Then query from 'Comedy_film' and other tables that you need to **find all the actors who acted in those comedy films**. The final output should be actors' id, first name, and last name only. (no duplicates and sort actor_id in descending order)

Submission format:

Create a file named "q2.sql"

Your sql file should look like this (if you miss "USE sakila;**" and "**DROP VIEW IF EXISTS Comedy_film;**" points will be deducted):**

```
USE sakila;
DROP VIEW IF EXISTS Comedy_film;

<your sql query>
```

<your sql query>

Solutions:

```
USE sakila;
DROP VIEW IF EXISTS Comedy_film;

CREATE VIEW Comedy_film AS
select c.category_id, c.name, fc.film_id
from category c
join film_category fc
on c.category_id = fc.category_id
where c.name = 'Comedy' ;

select distinct a.actor_id,a.first_name, a.last_name
from Comedy_film cf
join film_actor fa
on cf.film_id = fa.film_id
join actor a
on a.actor_id = fa.actor_id
order by a.actor_id desc;
```

3. [25 pts] Suppose one time you wish to find films that an actor played, but you couldn't remember the actor's full name. Instead, you only remember that his/her last name is "`Temple`". Luckily, you once created a table called '`nicer_but_slower_film_list`' in the `sakila` database where it stores all the information about films and actors.

However, a super villain named "Novie man" realized that that table still exists and cast a spell on your `mysql` command so that you can't use your `mysql` command at all. So every time you type `mysql`, your terminal spits out "`command not found`".

But you have Python! Use mysql.connector and write a python script called "search.py". Show what films (with fid) have an actor or actors whose name contains "Temple" (case-sensitive). In the meantime, show how many films you find.

Submission format:

a. Create a file named search.py
b. Don't print anything extra

c. Use “dsci551” as username and “Dsci-551” as password.

Execution format:

```
python search.py
```

Output format (print in terminal. First line is shown below, second line is an empty line, 3rd line and above are shown below, sorted by fid ascendingly):

```
37 films in total.

Anthony Temple plays A Beautiful Mind(1)
Cheryl Temple and Anthony Temple play Catch Me If You Can(5)
...
```

Note:

1. The word “and” between multiple actors
2. Verbs are different for singular/plural subjects
3. Title casing for the titles

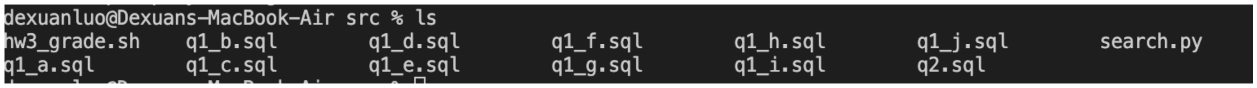
Solution:

```
import mysql.connector


config = {
        'user': 'dsci551',
        'password': 'Dsci-551',
        'database': 'sakila'
}

cnx = mysql.connector.connect(**config)
cursor = cnx.cursor()

query = ("select fid, title, actors, price from
nicer_but_slower_film_list where actors like '%Temple%';")

cursor.execute(query)
cursor.fetchall()
print(f"{cursor.rowcount} films in total.\n")
```

```
cursor.execute(query)

for (fid, title, actors, price) in cursor:
     actrs = [actor for actor in actors.split(', ') if 'temple' in
actor.lower()]
     print(f"{' and '.join(actrs)} play{'' if len(actrs)>1 else 's'}
{title.title()}({fid})")

cnx.close()
```

Submission:

1. Your submission folder should contain 13 files and look **EXACTLY** like this (PLEASE INCLUDES hw3_grade.sh, otherwise 10 pts will be deducted), any extra files like "README" will be ignored



```
dexuanluo@Dexuans-MacBook-Air src % ls
hw3_grade.sh    q1_b.sql        q1_d.sql        q1_f.sql        q1_h.sql        q1_j.sql        search.py
q1_a.sql        q1_c.sql        q1_e.sql        q1_g.sql        q1_i.sql        q2.sql
```

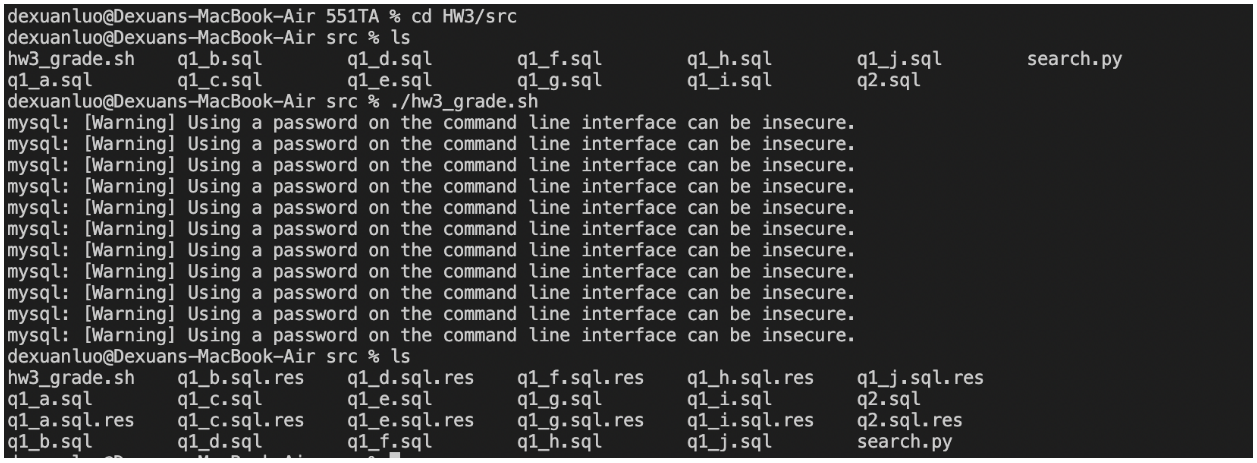
Please understand how TA will run your sql files for q1 and q2.

The TAs will simply run.

./hw3_grade.sh

And then the command will generate a bunch of ".res" files. Then TA will grade based on those ".res" files. If your filename is incorrect or your username and password is incorrect for the database points will be deducted. Test your files with the given grading script before you submit. **If you change a single byte in hw3_grade.sh, 50 pts will be deducted**.

After running the grading script your directory should look **EXACTLY** like



```
dexuanluo@Dexuans-MacBook-Air 551TA % cd HW3/src
dexuanluo@Dexuans-MacBook-Air src % ls
hw3_grade.sh    q1_b.sql        q1_d.sql        q1_f.sql        q1_h.sql        q1_j.sql        search.py
q1_a.sql        q1_c.sql        q1_e.sql        q1_g.sql        q1_i.sql        q2.sql
dexuanluo@Dexuans-MacBook-Air src % ./hw3_grade.sh
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
mysql: [Warning] Using a password on the command line interface can be insecure.
dexuanluo@Dexuans-MacBook-Air src % ls
hw3_grade.sh    q1_b.sql.res    q1_d.sql.res    q1_f.sql.res    q1_h.sql.res    q1_j.sql.res
q1_a.sql        q1_c.sql        q1_e.sql        q1_g.sql        q1_i.sql        q2.sql
q1_a.sql.res    q1_c.sql.res    q1_e.sql.res    q1_g.sql.res    q1_i.sql.res    q2.sql.res
q1_b.sql        q1_d.sql        q1_f.sql        q1_h.sql        q1_j.sql        search.py
```

2. Put all files in the same directory and compress it into a zip file.
   Zip file name format: LASTNAME_FIRSTNAME_HW3.zip

   Make sure when the file is unzipped, the folder name is LASTNAME_FIRSTNAME_HW3

3. If you modify a column or delete a record or drop a table from TA's database, your homework will be graded 0.