CS 377: Database Systems

Project #2 Solutions

SUBMISSION: Please submit the project electronically via Blackboard before 11:59 pm. Any submissions afterwards will be marked late and deducted accordingly unless you create a README file that specifies that you are using one of your 2 late assignment exceptions. You should use one file for each portion of the problem and follow these guidelines:

- If you have comments in your SQL query, use C style comments using /* < comment > */
- Any text that is not within the comment brackets will be assumed to be part of your SQL command, and if the file does not execute properly, points will be deducted.
- Each file should be named as "<netID>-project1-<problem>-<part>.sql". For example, the instructor's answer to part b of question 2 would have the file "jho31-project2-2-b.sql".
- At the top of each of your submitted SQL files should be the honor code.

```
/* THIS CODE IS MY OWN WORK.
IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS.
_Your_Name_Here_ */
```

1. **Online Bookstore** (40 points): This problem is based in the Online Bookstore exercise from Homework #1 and Project #1. Please include your creation and insertion queries into the first file so we can replicate your queries properly.

   (a) Find the total numbers of books that each costumer has bought of each publisher.

   (b) Update the ISBN number of one of the books and the email account of a customer you inserted. Verify that your updates were propagated to the other tables (no mismatches). Show the queries for the updates and the verifications (show the tables that reference these values with corresponding changes).

   (c) Delete one of the publishers that you have inserted that has published a book.

   (d) Create a view for the director of sales which contains a list of customers with their name, email, phone, the total number of books in their shopping cart (assuming that everybody orders just 1 copy of the book), and the total cost of their shopping carts.

   (**ANSWER**)

   (a)
   ```
   SELECT c.name, b.publiName, count(b.ISBN)
   FROM customer c, shopping_cart s, item_cart ic, book b
   WHERE c.email = s.cemail
   AND ic.cEmail=s.cEmail
   AND ic.cartSequence=s.cartSequence
   AND ic.ISBN = b.ISBN
   GROUP BY c.name, b.publiName;
   ```

   (b) The important thing for this problem is that the foreign keys have the keywords ON UPDATE CASCADE during creation (or you can alter the constraint). Otherwise, the updates will not work properly.

```
UPDATE customer
SET  email = 'joyce@emory.edu'
WHERE email = 'joyce.c.ho@emory.edu';
/* verify update to email gets pushed through */
SELECT *
FROM   item_cart;
SELECT *
FROM   shopping_cart;

UPDATE book
SET    ISBN = '5789'
WHERE  ISBN = '1234';
/* verify the update to the book isbn is good */
SELECT *
FROM author_book;
SELECT *
FROM item_cart;
```

(c) The important thing for this problem is that the foreign keys have the keywords ON DELETE CASCADE during creation (or you can alter the constraint). Otherwise, the delete will not work properly.

```
DELETE FROM book
WHERE publiName = 'Elsevier';
```

(d)
```
CREATE VIEW sales_director
AS (SELECT c.name, c.email, c.phone, COUNT(ic.ISBN), SUM(b.price)
FROM customer c, item_cart ic, book b
WHERE c.email = ic.cEmail AND b.ISBN = ic.ISBN
GROUP BY c.email);
```

2. **IMDB Database** (60 points): We will answer more sophisticated questions from our course's version of the IMDB database. Please see Project #1 and Piazza for the database details. Write SQL queries to answer the following questions. Each file should be named as "<netID>-project2-2-<part>.sql". For example, the instructor's answer to part b of this question would have the file "jho31-project2-2-b.sql".

   (a) (10 points) List all actors by their first and last name, sorted by the last name from A to Z, who have played in at least 10 different movies in 2004.

   (b) (10 points) List the movies and the year where Matt Damon, George Clooney, and Brad Pitt have all played a role.

   (c) (10 points) Who are the top 100 directors who have directed the most movies from 2005 to 2010, in descending order of the number of movies they have directed? Output their first name, last name, and number of movies directed.

   (d) (15 points) Which actor(s) has only been in Steven Soderbergh's movies?

   (e) (15 points) For each year, count the number of movies in that year that had only female actors (and at least one actor).

   (**ANSWER**)

   (a)
```
SELECT a.fname, a.lname, aid, COUNT(mid)
FROM movie m, casts c, actor a
WHERE year = 2004 AND m.id = c.mid AND a.id = c.aid
GROUP BY aid
HAVING COUNT(mid) >= 10;
```

(b) Use set intersection of the movie id between all three actors.

```
SELECT name, year
FROM movie
WHERE id IN
    ( SELECT mid
      FROM casts
      WHERE aid = (SELECT id FROM actor WHERE fname = 'Matt' AND lname = 'Damon') )
  AND id in
    ( SELECT mid
      FROM casts
      WHERE aid = (SELECT id FROM actor WHERE fname = 'George' AND lname = 'Clooney') )
  AND id in
    ( SELECT mid
      FROM casts
      WHERE aid = (SELECT id FROM actor WHERE fname = 'Brad' AND lname = 'Pitt') );
```

(c)
```
SELECT d.fname, d.lname, COUNT(md.mid)
FROM director d, movie_director md, movie m
WHERE md.mid = m.id AND m.year >= 2005 AND m.year <= 2010 AND d.id = md.did
GROUP BY md.did
ORDER BY COUNT(md.mid) DESC
LIMIT 100;
```

(d) One option is to use the set difference, so you first eliminate all the actors who have been in movies that were not directed by Steven Soderbergh. Once you have that, you want to make sure that the actors you have left have been a movie that was directed by Steven Soderbergh as there are movies that have no directors either.

```
SELECT *
FROM actor
WHERE id NOT IN
( SELECT aid
  FROM casts c, movie_director md, director d
  WHERE md.did = d.id AND md.mid = c.mid
     AND NOT (d.fname = 'Steven' AND d.lname = 'Soderbergh'))
AND id IN
( SELECT aid
  FROM casts c, movie_director md, director d
  WHERE md.did = d.id AND md.mid = c.mid
     AND d.fname = 'Steven' AND d.lname = 'Soderbergh');
```

(e) We can get the movies that have a male actor, and eliminate them from the movie list. Since we also need them to have at least one actor, we need to join it with the casts to make sure there is at least one non-null actor id. Finally we can group by the year and count the movies left in the list.

```
SELECT m.year, COUNT(DISTINCT m.id)
FROM movie m, casts c
WHERE m.id NOT IN
    (SELECT m.id
     FROM movie m, actor a, casts c
     WHERE m.id = c.mid AND a.id = c.aid AND a.gender = 'M')
    AND m.id = c.mid
GROUP BY m.year;
```