CSCD 327 Lab #6 (17 points + 3 extra)
Due: 12pm on Nov 14, 2011

Part I: Database Updates

**** Important: Create a new database named *YourUsername_4_2*. Import the tables from the file *database4.sql* into *YourUsername_4_2* database. Use *YourUsername_4_2* for the following exercises.**

Write the following queries in **SQL** statements.

1. Increase the salary of each instructor in the Comp. Sci. department by 10%.
2. Delete all courses that have never been offered (that is, do not occur in the *section* relation).
3. Insert every student whose *tot_cred* attribute is greater than 100 as an instructor in the same department, with a salary of $10,000.
4. Create a new course "CS-001", titled "Weekly Seminar", with 0 credits.
5. Create a section of this course in Fall 2009, with *section id* of 1.
6. Enroll every student in the Comp. Sci. department in the above section.
7. Delete enrollments in the above section where the student's name is Chavez.
8. Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course?
9. Delete all *takes* tuples corresponding to any section of any course with the word "database" as a part of the title; ignore case when matching the word with the title.

Part II: Views

Use *YourUsername*_2 database for the following exercises.

10. List all products, with MFR_ID, PRODUCT_ID, and salesperson's ID and the total quantity of sales (named as Total_qty_by_salesperson). The result should include the products that have never been ordered. **Include the query and the number of rows returned by the query in your answer**.
11. Write a SQL statement to create a view called *AllProductsSales* based on the query in Exercise 10.
12. **Important**: The view does not store any data. The data is stored just in the tables used in the definition of view. When a query that uses *AllPorductsSales* is executed, the system first evaluates the query that defines *AllProductsSales*, and then performs further evaluation of the query that uses the view.

    To demonstrate this, do the following:

    a. Insert a new row ('AAA', '44444', 'New Product', 100, 10) into PRODUCTS table:

insert into PRODUCTS values ('AAA', '44444', 'New Product', 100, 10);

b. Execute the query from exercise 12 again.
c. Did the results changed? Why?

13. Run the following query (or the similar query based on your data):

    UPDATE AllProductsSales
    SET rep = 106
    WHERE rep IS NULL;

    Does the query succeed? If not, copy the error message, and explain why you received this error message.
14. **(Extra 1 point)** Use *AllProductsSales* to create another view called *ProductProfit* to show the MFR_ID, PRODUCT_ID, and Total_profit of the product. Total_profit = total number of orders * PRICE * 0.1 (this assumes the profit is 10%).
15. **(Extra 1 point)** SQL SELECT statement for MySQL has a "LIMIT *n*" clause that allows you to limit the number of rows returned in the result of a select query. For example "SELECT * FROM PRODUCTS LIMIT 5" will display first 5 rows from Products table. Write the SQL query on the *ProductProfit* view to show the top 5 products based on total profit in order of highest profit to lowest profit.
16. **(Extra 1 point)** Write the SQL statement (using the view ProductProfit) to update the Total_profit for (MFR_ID = 'ACI' and PRODUCT_ID = 41001) to be $100,000. Run the query. Does the query succeed? Why?

Part III: Security

As a database user, you currently have a username, password, and you have the rights (permissions) to a particular database (e.g., *YourUsername*_1) where you can create tables, insert into them, etc. Assume now that you are a database administrator, and you are the one giving permissions to users to use the database. For security reasons, you should restrict access to your database as much as possible, and only give permissions that are really needed in order to accomplish the task. We'll accomplish this through the use of the SQL GRANT statement.

GRANT <permissions>
[ON <table(s)>]
TO <user>

For example, *GRANT INSERT ON YourUsername_1.EMP to danl* will allow the user *danl* to insert data into the *EMP* table in the database called *YourUsername_1*.

Now, let's take a closer look at the GRANT statement line-by-line. The first line, GRANT <permissions>, allows us to specify the specific permissions we are granting. These can be either table-level permissions (such as SELECT, INSERT, UPDATE and DELETE) or database permissions (such as CREATE TABLE, DROP TABLE, ALTER DATABASE, GRANT, etc). More than one permissions can be granted in a single GRANT statement, but table-level permissions and database-level permissions may not be combined in a single statement.

The second line, ON <table>, is used to specify the affected table for table-level permissions. Use the format *databaseName.tableName(s).* This line is omitted if we are granting database-level permissions.

The third line specifies the user that is being granted permissions.

For this part of the lab, you will need a partner. Please work with a classmate to accomplish and test the tasks below:

17. Ask your partner to log into phpMyAdmin and select all employees from your *EMP* table: SELECT * FROM ***YourUsername_1.EMP***. What is the answer?
18. Write a GRANT statement to grant SELECT privileges on your *EMP* table to your partner. Ask your partner to: 1) log out phpMyAdmin, 2) log back in into phpMyAdmin, and then 3) run the SELECT * FROM *YourUsername_1.EMP*. What is the answer this time?
19. What happens when your partner tries to INSERT a new employee into your database by using the following statement: INSERT INTO EMP VALUES(9999, 'SuperMan', 'PRESIDENT', NULL, '2000-01-01', 100000, NULL, 10)? Why?
20. Now revoke the privileges that you previously granted to your partner: REVOKE SELECT ON ***YourUsername_1.EMP*** FROM *YourPartnersUsername.* Ask your partner to: 1) log out phpMyAdmin, 2) log back in into phpMyAdmin, and then 3) run the SELECT * FROM ***YourUsername_1.EMP***. What is the answer this time?