SY306 Lab Nine

SQL Basic

In this lab you will develop your database implementation skills by creating and modifying tables in a MySQL database, and by inserting, deleting, modifying, and querying data using SQL.

Preliminaries

1. Connect to the MySQL server on csmidn and remove any tables that already exist. (MySQL instructions are available in course resources and here.) **Note:** You will use the command line client to test all the SQL statements for this lab. Each statement must end with a **;**

2. You must create a folder in your `public_html` called "Lab09" (without the quotes) and store your work in that directory.

3. Create all of your answers within a text file named *`yourlastname_lab09.sql`*

4. Follow **exactly** the following format for your file:

```
# Name, alpha
# {1}
/* question 1 here */
Answer 1 here

/* other comments if desired */

# {2}
/* question 2 here */
Answer 2 here

...
```

5. **Your format must be what is shown above.** Do not add any letters (such as 1A), use only the problem number.

6. Add comments as needed with a **#**, -- , or multi line comments can be enclosed in a C style **/*** ---- ***/** fashion.

7. **Hint:** You can test your work by running your sql file from the command line!

The actual lab:

Important Note: Your queries should run correctly on any data, not just the sample data currently in the database.

Write all your answers in the *`yourlastname_lab09.sql`* file, in the format specified above.

Part A:

Write the SQL statements to accomplish the following tasks:

1. Create the following 3 tables for your online store. The primary keys are underlined, and the foreign keys are shown in *italic*. You can personalize the tables as needed to better store the information you need for your website's products.

    i. **PRODUCT**(ProductID, PName, PDescription, Price) this table records information about each product. The table must have a numeric primary key.

    ii. **SALE**(SaleID, DeliveryAddress, CreditCard) this table records information about each sale/transaction. SaleID must be a surrogate key created by MySQL. Hint: Declare *SaleID int AUTO_INCREMENT* in the column declaration for SaleID.

    iii. **SALEITEM**(*SaleID, ProductID*, Quantity) this table records information about which products were sold in each sale/transaction. You don't have to specify the foreign keys at this point, only the primary key. Hint: The primary key can be a combination of columns.

2. Insert 3 products in your PRODUCT table.

3. Update the price of one of your products to cost $10 more than original price.

4. Delete all products with price higher than $10,000 from your PRODUCT table. Hint: You can insert such a product first and then use this command to delete it for testing purposes

5. Modify the SALEITEM table (using ALTER) to add the foreign key constraints for SaleID and ProductID.

6. Write the SQL query to list the SaleID, ProductID, PName, PDescription, Price, Quantity and DeliveryAddress for all sales with delivery address containing the string 'MD' anywhere in the address. The results should be displayed in ascending order by SaleID and ProductID.

Part B:

For this part, you will work on the Madison Importing database: Madison Importing purchases antiques and home furnishings in Asia and ships those items to a warehouse facility in Los Angeles. A database is used to keep track of items purchased, shipments and items shipped. The following tables, with primary key underlined and foreign keys in *italic* are part of the database:

- **ITEM**(ItemID, Description, PurchaseDate, Store, City, Weight, PriceInLocalCurrency, ExchangeRate) - this table stores information about the items purchased, including the weight and purchase price
- **SHIPMENT**(ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate, InsuredValue) - this table stores information about shipments, including the total value for which the shipment was insured
- **SHIPPED_ITEM**(*ShipmentID*, ShipmentItemNb, *ItemID*, Value) - this table stores information about the items included in each shipment, including the expected selling value of each item shipped

Download the `createMadisonImports.sql` file from the this **link**, save it to your Lab09 folder, and copy-paste the contents into the MySQL command line client to execute the entire file. It should create the tables for Madison Importing, with some data in them.

Now, write **SQL queries** to accomplish the following tasks.

7. Write the following three queries:

    a. List all data from the ITEM table.

    b. List all data from the SHIPMENT table

    c. List all data from the SHIPPED_ITEM table

8. List all the cities from the ITEMS table. Each city should only appear once in the result (no duplicates)

9. List the ItemID, Description, Store, City and Weight for items with weight between 2 and 50, in ascending order by City, and descending by Weight.

10. List all Stores where an item with expected selling value (from SHIPPED_ITEM) higher than 0900 was purchased. Each store should appear only once in the result.

11. List theShipmentID,ShipperName, ShipperInvoiceNumber, and InsuredValue for all shippers whose name starts with 'AB' and the insured value is greater than $13000.

12. List theShipperNameand DepartureDate of all shipments that have an item that was purchased in Singapore. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.

13. List theShipmentID,ShipperName, ItemID, Description, and City for all shipped items purchased in Singapore or Manila.

14. (Extra credit) For all items, showItemID, Description, Store and a calculated column named USCurrencyAmount that is equal to the PriceInLocalCurrency multiplied by ExchangeRate. Hint: look at the examples of using SQL Alias at http://www.w3schools.com/sql/sql_alias.asp

15. (Extra Credit) List theShipmentID of shipments that have at least two items: one purchased in Singapore and one purchased in Manila. Display the results sorted by ShipmentID. Hint:

    i. any condition that specifies City = "Singapore" AND City = "Manila" is always false

    ii. You can specify the same table multiple times in the FROM clause (you can join a table with itself), just rename them

16. (Extra Credit) List the maximum weight for an item shipped after '2009-01-01' (DepartureDateafter '2009-01-01').

To help you with debugging, below are the results I obtained from running the queries in Part B on the sample data. You should get something similar, but obtaining the same results as I did on the sample data does not guarantee that your SQL is correct. ***Your SQL queries should work for ANY possible data in the database.***

Click 🔍 icon to reveal query results.

Deliverables:

1. Your file *yourlastname_lab09.sql* file containing all the questions and the SQL statements should be in your Lab09 folder.

2. In your default.html page, under the heading Lab09 create a link to Lab09*/ yourlastname_lab09.sql*

3. **Turn in (due before lab on Friday after spring break):**

    a. **Paper submission:** turn in the following at the beginning of Lab10, stapled together in the following order (coversheet on top):

        i. A completed assignment coversheet. Your comments will help us improve the course.

        ii. A printout of *yourlastname_lab09.sql* file containing all the questions and the SQL statements

    b. **Electronic submission:** Submit the Lab09 SQL file [`yourlastname_lab09.sql`] via the online system: `submit.cs.usna.edu` by the following deadlines:

        - Section **4341**: 23:59, Wednesday, March 20th
        - Sections **1151, 3351, 5551**: 23:59, Thursday, March 21st

4. **NOTE:** The SQL solutions in your report will be autograded. Proper syntax for `yourlastname_lab09.sql` file is **KEY!** Before submitting, test your work by running your sql file from the command line!