Timed Lab 4 – SQL Data Manipulation

This is a Timed Lab; this Timed Lab is worth 20 **Exam** points.

For this Timed Lab, you *may* use	However, you *may not*
• Course notes • Homeworks • Recitation assignments • Other course material • Any material you may find on the Internet that don't involve communicating "live" with other people.	• Communicate with other people/students in real-time via any means. This means no Facebook, email, Piazza, IM, IRC, cell phones, Google Talk, smoke signals, etc. • Share code with other students. • Look at other students work.

The TAs will be available to answer clarifying questions about the problem, but they are not permitted to give assistance with debugging code, program logic, etc. You will have an entire recitation period to work on this assignment; this time begins *exactly* when your recitation begins and ends *exactly* when your recitation ends: No extra time will be given if you arrive late, except in highly extenuating circumstances that must be approved by Dr. Summet.

T-Square will not permit any late submissions; ensure that you submit your code to T-Square several times to prevent earning a zero due to you being unable to submit. Your TAs will give a verbal warning 10 and 5 minutes before the end of the recitation period; you should submit at these times. If you are taking this timed lab out of section (with approval from Dr. Summet or the Head TA), please e-mail your code to your grading TA by the end of the recitation you are in.

In your collaboration statement, if you use code from somewhere that is *not* a class resource (i.e. not listed on the course calendar), please list where this code came from. Ensure that you fill out the header at the top of the file.

Problem Description:

In this timed lab, you will write a series of four functions that will test your knowledge of PyMySQL, SQL query writing, and general Python knowledge. You will not be writing a GUI for this timed lab. You will be using the **demand** table located on the class database for the entirety of this assignment. For these assignments, you may assume that your connection to the database will always succeed, and you do not need to error check for this. *Each function will need to make its own connection to the database.*

Function 1 -- updateDate:

Write a function in Python named **updateDate** that takes in three parameters: A date (a string of the format YYYY-MM-DD) and a product name, and a demand amount. You will then update any row in the database that matches the given date and name to have the given demand. You should not return anything.

Function 2 -- getDates:

Write a function in Python named **getDates** that takes in no parameters and returns a list of all the unique dates in the database. The dates must be returned as strings. You may accomplish selecting the unique dates either through the SQL query or filtering out the duplicate dates in Python.

Function 3 -- addDemandRow:

Write a function in Python named addDemandRow that accepts no parameters and will return nothing. This function will ask the user for input (using the console/text based input function), requesting that they enter the data for the new row. The data will be typed in by the user as one string, which will contain three pieces of data, in the following order:

1. **The date, in YYYY-MM-DD format**
2. **The product – Apple, Pear, Grape**
3. **The demand – an integer**

Each piece of data will be separated by two forward slashes, with a space on both sides. For example:

2009-12-01 // Apple // 24

You should display a prompt asking the user to enter their string, and then check to see if they entered a string which was properly formatted, and contains all three elements (HINT: Regex may be helpful!). If the user did not enter a valid string (i.e. they gave a negative demand, they didn't list one of the valid products, or they didn't provide all three pieces of data), display an error message and quit the function. If the user did enter a valid string, you will add a new row to the database, and then print a message saying the row was successfully added.

You do not need to check the date for validity, but must check it for proper formatting: If the date the user enters matches the format YYYY-MM-DD, you may assume that this is a valid date. Note that Y, M, and D are all integers.

Grading:

updateDate:

+1: Successfully connects to database and gets cursor
+3: Correctly updates date
+1: Closes database connection and cursor

getDates:

+1: Successfully connects to database and gets cursor
+1: Gets list of dates
+2: Returns list with only unique dates
+1: Closes cursor and database connection

addDemandRow:

+1: Successfully connects to database and gets cursor
+2: Asks for user input with correct prompt
+2: Ensures that input contains three pieces of data
+2: Checks individual pieces of data for validity
+2: Correctly inserts into database (including commit)
+1: Closes database connection and cursor