CPSC 3300 – Lab 4

<u>Objectives</u>

By the end of the lab you will be able to:

- Use transactions
- Create stored procedures

<u>Lab Assignment</u>

1. (10 points)
   Download the MySQL script "mysqlsampledatabase.sql" from Canvas and load the script into your database on cssql.

   This sample database consists of the following tables:
   - **Customers**: stores customer's data
   - **Products**: stores a list of scale model cars
   - **ProductLines**: stores a list of product line categories
   - **Orders**: stores sales orders placed by customers
   - **OrderDetails**: stores sales order line items for each sales order
   - **Payments**: stores payments made by customers based on their accounts
   - **Employees**: stores all employee information as well as the organization structure such as who reports to whom
   - **Offices**: stores sales office data

We want to add a new sale order for the customer (customerNumber = 145) in the database. The steps of adding a sale order are described as follows:

1) Get latest sale order number from "orders" table, and use the next sale order number as the new sale order number
2) Insert a new sale order into "orders" table for the customer (customerNumber = 145). For this order, the orderNumber is the new sale order number from step 1), orderDate is the current date (you can use now() to get the date), requiredDate is 5 days from now (you can use date_add(now(), INTERVAL 5 DAY) to get the date), shippedDate is 2 days from now (you can use date_add(now(), INTERVAL 2 DAY) to get the date), status is "in process".
3) Insert new sale order items into "orderdetails" table. The customer has bought two items in his order. One item has productCode = 'S18_1749', quantityOrdered = 30, priceEach for this item is 136, orderLineNumber = 1. The second item has productCode = 'S18_2248', quantityOrdered = 50, priceEach for this item is 55.09, orderLineNumber = 2.

Write a MySQL script to start a transaction to ensure that the database never contains the result of partial operations above.

2. (3 points)
   Use the database from question 1 to solve the problem below.

Create a stored procedure “setRelocationFee”to set the relocation fee for a given employee. If the employee’s office is in San Francisco, the relocation fee is $10000; if the employee’s office is in Boston, the relocation fee is $8000; if the employee’s office is in London, the relocation fee is $20000; if the employee works in other offices, the relocation fee is $15000.

Below is a sample statement to test your stored procedure.

```
set @employeeID = 1501;
call setRelocationFee(@employeeID, @relocationfee);
select @employeeID, @relocationfee;
```

3. (3 points)
   Use the database from question 1 to solve the problem below.

   Create a stored procedure “changeCreditLimit” to change the credit limit for a given customer. If the customer’s total payment amount (note: payment amount is in the table “payments”) is not smaller than a given amount, then add 2000 to the customer’s credit limit (note: credit limit is in the table “customers”).

   Below is a sample statement to test your stored procedure.

   ```
   set @customer = 114;
   set @totalpayment = 15000;
   call changeCreditLimit(@customer,@totalpayment);
   ```

   The current credit limit of customer “114” is 117300. After the procedure “changeCreditLimit” is invoked, the customer’s credit limit should become 119300.

4. (4 points)

Create a table using the statement below.

```
create table odd (number int primary key);
```

Then create a stored procedure “insertOdd” to insert odd numbers in the range of [1, maxRange] into the table “odd”. If the number 5 and/or 15 appear in the range, they are skipped.

Below are sample statements to test your stored procedure.

```
set @maxRange;
call insertOdd(maxRange);
```

<u>Submission</u>

Use Canvas to submit your SQL statements (in a Word file or .sql file). Make sure you test your SQL statements on the database server “cssql.seattleu.edu” before submission. I will grade your SQL statements by running them on the server.