CSCI 5020 Assignment 5

Part 1

Chapter 14 Exercises 1, 2, 3 on page 445.

Chapter 15 Exercises 1, 2, 3, 4, 6, 7, 8 on page 497 - 498.

Chapter 16 Exercises 1, 2, 3 on page 522.

Chapter 17 Exercises 1 on page 548.

Chapter 18 Exercises 1, 2, 3, 6 on page 599.

Chapter 19 Exercises 1, 2, 3, 4 on page 631.

The solutions to the chapter exercises in the book are provided through the student download and posted in D2L as well. Do not turn in this part for grading.

Part 2

Chapter 14 How to code scripts

1. Write a script that declares a variable and sets it to the count of all products in the Products table. If the count is greater than or equal to 7, the script should display a message that says, “The number of products is greater than or equal to 7”. Otherwise, it should say, “The number of products is less than 7”.
2. Write a script that uses two variables to store (1) the count of all of the products in the Products table and (2) the average list price for those products. If the product count is greater than or equal to 7, the script should print a message that displays the values of both variables. Otherwise, the script should print a message “The number of products is less than 7”.

Chapter 15 How to code stored procedures, functions, and triggers

2. Write a script that creates and calls a function named fnDiscountPrice that calculates the discount price of an item in the OrderItems table (discount amount subtracted from item price). To do that, this function should accept one parameter for the item ID, and it should return the value of the discount price for that item.

4. Write a script that creates and calls a stored procedure named spInsertProduct that inserts a row into the Products table. This stored procedure should accept five parameters. One parameter for each of these columns: CategoryID, ProductCode, ProductName, ListPrice, and DiscountPercent.

   This stored procedure should set the Description column to an empty string, and it should set the DateAdded column to the current date.

   If the value for the ListPrice column is a negative number, the stored procedure should raise an error that indicates that this column doesn’t accept negative numbers. Similarly, the procedure should raise an error if the value for the DiscountPercent column is negative.

   Code at least two EXEC statements that test this procedure.

6. Create a trigger named Products_UPDATE that checks the new value for the DiscountPercent column of the Products table. This trigger should raise an appropriate error if the discount percent is greater than 100 or less than 0.

   If the new discount percent is between 0 and 1, this trigger should modify the new discount percent by multiplying it by 100. That way, a discount percent of .2 becomes 20.

   Test this trigger with an appropriate UPDATE statement.

Chapter 16 How to work with cursors

1. Write a script that creates a cursor for a result set that consists of the ProductName and ListPrice columns for each product with a list price that's greater than $700. The rows in this result set should be sorted in descending sequence by list price. Then, the script should print the product name and list price for each product so it looks something like this:

   ```
   Gibson SG, $2517.00
   Gibson Les Paul, $1199.00
   ```

2. Write a script to declare and use a cursor for the following SELECT statement. Use a WHILE loop to fetch each row in the result set. Omit the INTO clause to fetch directly to the Results tab.

   ```
   SELECT LastName, AVG(ShipAmount) AS ShipAmountAvg
   FROM Customers JOIN Orders
       ON Customers.CustomerID = Orders.CustomerID
   GROUP BY LastName;
   ```

Chapter 17 How to work with transactions and locking

1. Write a script that includes two SQL statements coded as a transaction to delete the row with a customer ID of 8 from the Customers table. To do this, you must first delete all addresses for that customer from the Addresses table.

   If these statements execute successfully, commit the changes. Otherwise, roll back the changes.

2. Write a script that includes these statements coded as a transaction:

   ```
   INSERT Orders
   VALUES (3, GETDATE(), '10.00', '0.00', NULL, 4,
     'American Express', '378282246310005', '04/2013', 4);

   SET @OrderID = @@IDENTITY;

   INSERT OrderItems
   VALUES (@OrderID, 6, '415.00', '161.85', 1);

   INSERT OrderItems
   VALUES (@OrderID, 1, '699.00', '209.70', 1);
   ```

   Here, the @@IDENTITY variable is used to get the order ID value that's automatically generated when the first INSERT statement inserts an order.

   If these statements execute successfully, commit the changes. Otherwise, roll back the changes.

Chapter 18 How to manage database security

1. Write a script that creates a user-defined database role named OrderEntry in the MyGuitarShop database. Give INSERT and UPDATE permission to the new role for the Orders and OrderItems table. Give SELECT permission for all user tables.
2. Write a script that (1) creates a login ID named “RobertHalliday” with the password “HelloBob”; (2) sets the default database for the login to the MyGuitarShop database; (3) creates a user named “RobertHalliday” for the login; and (4) assigns the user to the OrderEntry role you created in exercise 1.
3. Write a script that uses dynamic SQL and a cursor to loop through each row of the Administrators table and (1) create a login ID for each row in that consists of the administrator’s first and last name with no space between; (2) set a temporary password of “temp” for each login; (3) set the default database for the login to the MyGuitarShop database; (4) create a user for the login with the same name as the login; and (4) assign the user to the OrderEntry role you created in exercise 1.
4. Using the Management Studio, create a login ID named “RBrautigan” with the password “RBra9999,” and set the default database to the MyGuitarShop database. Then, grant the login ID access to the MyGuitarShop database, create a user for the login ID named “RBrautigan”, and assign the user to the OrderEntry role you created in exercise 1.

   Note: If you get an error that says “The MUST_CHANGE option is not supported”, you can deselect the “Enforce password policy” option for the login ID.

Chapter 19 How to work with XML

1. Write a SELECT statement that returns an XML document that contains all of the current shipping addresses for the customers in the database. This document should include one element for each of these columns: FirstName, LastName, Line1, Line2, City, State, and ZipCode. Then, save the XML document that’s returned in a file named CustomerAddresses.xml. Finally, generate an XML schema for the file and save it in a file named CustomerAddresses.xsd.
2. Write a script that inserts a new row into the Customers table for each customer stored in the NewCustomers.xml file that’s in the Exercise Starts directory.

   To accomplish this, begin by storing the contents of the XML file in a variable of the XML type. Then, you can use an INSERT statement to insert the rows into the Customers table.
3. Write a script that returns a result set that contains all of the data stored in the NewCustomers.xml file that’s in the Exercise Starts directory.

   To accomplish this, begin by storing the contents of the XML file in a variable of the XML type. Then, you can use a SELECT statement to return the data stored in this variable as a result set.

Turn in screenshot for Chapter 18 Exercise 4. Write each statement in separate script files (*.sql) for exercises of other chapters. Zip all the files into a single file and submit it to the Dropbox.