CSCI 5020 Assignment 3

Part 1

Chapter 6 Exercises 2, 4, and 6 on page 213.

Chapter 7 Exercises 3, 6, 7, and 9 on page 236 - 237.

Chapter 8 Exercises 1 on page 260.

Chapter 9 Exercises 2, 4, and 5 on page 300.

The solutions to the chapter exercises in the book are provided through the student download and posted in D2L as well. For learning purpose, make sure you complete each exercise before checking the solution. You do not need to turn in this part for grading.

Part 2

Chapter 6 How to code subqueries

1. Write a SELECT statement that returns the same result set as this SELECT statement, but don't use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword.

```
SELECT DISTINCT CategoryName
FROM Categories c JOIN Products p
  ON c.CategoryID = p.CategoryID
ORDER BY CategoryName
```

3. Write a SELECT statement that returns the CategoryName column from the Categories table.

   Return one row for each category that has never been assigned to any product in the Products table. To do that, use a subquery introduced with the NOT EXISTS operator.

4. Write a SELECT statement that returns three columns: EmailAddress, OrderID, and the ordertotal for each customer. To do this, you can group the result set by the EmailAddress and OrderIDcolumns. In addition, you must calculate the order total from the columns in the OrderItems table.

   Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer's email address and the largest order for that customer. To do this, you can group the result set by the EmailAddress column.

Chapter 7 How to insert, update, and delete data

To test whether a table has been modified correctly as you do these exercises, you can write and run an appropriate SELECT statement.

1. Write an INSERT statement that adds this row to the Categories table:

   CategoryName: Brass

   Code the INSERT statement so SQL Server automatically generates the value for the CategoryID column.

2. Write an UPDATE statement that modifies the row you just added to the Categories table. This statement should change the CategoryNamecolumn to “Woodwinds”, and it should use the CategoryIDcolumn to identify the row.
3. Write a DELETE statement that deletes the row you added to the Categories table in exercise 1. This statement should use the CategoryID column to identify the row.
7. Write an INSERT statement that adds this row to the Customers table:

EmailAddress:	rick@raven.com
Password:	(empty string)
FirstName:	Rick
LastName:	Raven

   Use a column list for this statement.
8. Write an UPDATE statement that modifies the Customers table. Change the password column to “secret” for the customer with an email address of rick@raven.com.
10. Open the script named CreateMyGuitarShop.sql that’s in the Exercise Starts directory. Then, run this script. That should restore the data that’s in the database.

Chapter 8 How to work with data types

1. Write a SELECT statement that returns these columns from the Productstable:

   The ListPrice column

   A column that uses the CAST function to return the ListPrice column with 1 digit to the right of the decimal point

   A column that uses the CONVERT function to return the ListPrice column as an integer

   A column that uses the CAST function to return the ListPricecolumn as an integer
3. Write a SELECT statement that returns these colums from the Orders table:

   A column that uses the CONVERT function to return the OrderDate column in this format: MM/DD/YYYY. In other words, use two-digit months, days, and years and separate each date component with slashes

   A column that uses the CONVERT function to return the OrderDate column with the date, and the hours and minutes on a 12-hour clock with an am/pm indicator

   A column that uses the CONVERT function to return the OrderDate column with 2-digit hours, minutes, and seconds on a 24-hour clock. Use leading zeros for all date/time components.

Chapter 9 How to use functions

2. Write a SELECT statement that returns these columns from theOrders table:

   The OrderDate column

   A column that returns the four-digit year that’s stored in the OrderDate column

   A column that returns only the day of the month that’s stored in the OrderDate column.

   A column that returns the result from adding thirty days to the OrderDate column.

3. Write a SELECT statement that returns these columns from the Orderstable:

   The CardNumber column

   The length of the CardNumber column

   The last four digits of the CardNumbercolumn

   When you get that working right, add the column that follows to the result set. This is more difficult because the column requires the use of functions within functions.

   A column that displays the last four digits of the CardNumber column in this format: XXXX-XXXX-XXXX-1234. In other words, use Xs for the first 12 digits of the card number and actual numbers for the last four digits of the number.

4. Write a SELECT statement that returns these columns from theOrders table:

   The OrderID column

   The OrderDate column

   A column named ApproxShipDate that's calculated by adding 2 days to the OrderDate column

   The ShipDate column

   A column named DaysToShip that shows the number of days between the order date and the ship date

   When you have this working, add a WHERE clause that retrieves just the orders for March2012.

Write each statement in separate script files (*.sql). Zip all the script files into a single file and submit it to the Dropbox.