



## Project 1

### I. My Guitar Shop Database

In part I, you will use SQL Server Management Studio to create the MyGuitarShop database, to review the tables in the MyGuitarShop database, and to enter SQL statements and run them against this database.

#### A Database Setup [2 pts.]

1. (1) Download CreateMyGuitarShop.sql from Project 1 directory on Blackboard and open it in SQL server management studio. Execute the entire script and show the message in the Message tab, indicating the script is executed successfully. A complete screenshot of execution result is required.  
(2) Navigate through the database objects and view the column definitions for each table. Open a new Query Editor window. Show details in Categories table and Products table using SELECT statement. Full screenshots of execution results are required.

#### B An introduction to SQL [6 pts.]

1. [3] Write a SELECT statement that returns one column from the Addresses table named Address that joins the State and ZipCode columns.

Format this column with the State, a comma, a space, and the ZipCode like this:

NY, 13210

Add an ORDER BY clause to this statement that sorts the result set by State in ascending sequence. Return only the addresses whose state names begin with a letter from A to F.

2. [3] Write a SELECT statement that returns three columns: OrderID, OrderDate, ShipDate from the Orders table. Return only the rows where the ShipDate column contains a null value.

## C The essential SQL skills [44 pts.]

1. [4] Write a SELECT statement that joins the Customers, Orders, OrderItems, and Products tables. This statement should return these columns: FirstName, OrderDate, ProductName, ItemPrice, DiscountAmount, and Quantity. Use aliases for the tables. Sort the final result set by FirstName in descending order, and in ascending order for OrderDate, and ProductName.
2. [4] Write a SELECT statement that selects CategoryName column from the Categories table. Return one row for each category that has been used. (Hint: Use an outer join and only return rows where the ProductID column does not contain a null value).
3. [4] Write a SELECT statement that returns one row for each customer that has orders with these columns:
  - a) The EmailAddress column from the Customers table
  - b) The average of the item price in the OrderItems table multiplied by the quantity in the OrderItems table
  - c) The average of the discount amount column in the OrderItems table multiplied by the quantity in the OrderItems table.Sort the result set in ascending sequence by the item price average for each customer.
4. [4] Write a SELECT statement that returns one row for each customer that has orders with these columns:
  - a) The EmailAdress column from the Customers table
  - b) A count of the number of orders
  - c) The total amount for each order (Hint: First, subtract the discount amount from the price. Then, multiply by the quantity)Return only those rows where items have a more than 700 ItemPrice value.  
Sort the result set in ascending sequence by the sum of the line item amounts.
5. [4] (1) Write a SELECT statement that returns three columns: EmailAddress, OrderID, and the order average for each customer. To do this, you can group the result set by the EmailAddress and OrderID columns. In addition, you must calculate the order average from the columns in the OrderItems table.  
(2) 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 average for that customer. To do this, you can group the result set by the EmailAddress column.
6. [4] Use a correlated subquery to return one row per customer, representing the customer's oldest order (the one with the oldest date). Each row should include these three columns: EmailAddress, OrderID, and OrderDate.
7. [4] Write a SELECT statement that returns these columns from the Products table:
  - a) The ListPrice column
  - b) A column that uses the CAST function to return the ListPrice column with 3 digits to the right

- of the decimal point
- c) A column that uses the CONVERT function to return the ListPrice column as a real number
  - d) A column that uses the CAST function to return the ListPrice column as an integer.
8. [4] Write a SELECT statement that returns these columns from the Orders table:
- a) The CardNumber column
  - b) The length of the CardNumber column
  - c) The last three digits of the CardNumber column
  - d) 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.
9. [4] Write a SELECT statement that returns these columns from the Orders table:
- a) The OrderID column
  - b) The OrderDate column
  - c) A column named ApproxShipDate that's calculated by adding 3 days to the OrderDate column
  - d) The ShipDate column
  - e) A column named MonthsToShip that shows the number of months between the order date and the ship date

When you have this working, add a WHERE clause that retrieves just the orders for Jan to June 2016.

For question 10-11: To test whether a table has been modified correctly as you do these questions, please write and run an appropriate SELECT statement and take full screenshots of the verification.

10. [4] Write an INSERT statement that adds this row to the Customers table:

EmailAddress: [noorie@krieger.com](mailto:noorie@krieger.com)

Password: (empty string)

FirstName: Noorie

LastName: Krieger

Use a column list for this statement.

11. [4] Write an UPDATE statement that modifies the Customers table. Change the password column to "secret" for the customer with an email address of [noorie@krieger.com](mailto:noorie@krieger.com).

**D Advanced SQL skills (views/stored procedures/functions/scripts) [24 pts.]**

Open the script named CreateMyGuitarShop.sql and run this script. That should restore the data that's in the database. Then complete questions in Section D. Screenshot of execution is required for each question. Please also use SELECT statement to verify results of your codes (Screenshots of verification are required).

1. [4] Create a view named OrderItemProducts that returns columns from the Orders, OrderItems, and Products tables.
  - a) This view should return these columns from the Orders table: OrderID, OrderDate, TaxAmount, and ShipDate.
  - b) This view should return these columns from the OrderItems table: DiscountAmount, FinalPrice (the discount amount subtracted from the item price), Quantity, and ItemTotal (the calculated total for the item).
  - c) This view should return the ProductName column from the Products table.
2. [5] Create a view named Top5LeastSelling that uses the view you created in Section D Question 1. This view should return some summary information about five least selling products. Each row should include these columns: ProductName, OrderTotal (the total sales for the product) and OrderCount (the number of times the product has been ordered).
3. [5] Write a script that creates and calls a stored procedure named spUpdateProductDiscount that updates the DiscountPercent column in the Products table. This procedure should have one parameter for the product ID and another for the discount percent.  
If the value for the DiscountPercent column is a negative number, the stored procedure should raise an error that indicates that the value for this column must be a positive number.  
Code at least two EXEC statements that test this procedure.
4. [5] Write a script that calculates the common factors between 8 and 24. To find a common factor, you can use the modulo operator (%) to check whether a number can be evenly divided into both numbers. Then, this script should print lines that display the common factors like this:

Common factors of 8 and 24

1  
2  
4  
8

5. [5] (1) 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. (2) Write a script that creates and calls a function named fnItemTotal that calculates the total amount of an item in the OrderItems table (discount price multiplied by quantity). To do that, this function should accept one parameter for the item ID, it should use the DiscountPrice function that you created in (1), and it should return the value of the total for that item.

## II. Database Design [20 pts.]

Create a sample database design for Cuse's Box Office for all Sports matches (Basketball, Soccer, Lacrosse, etc) and draw one database model for it. The Box Office needs to keep a track of all types of sports matches taking place at the Dome as well as all other 'away' locations.

By tracking the type of sport, its location, and the teams (and their captains) involved per match, the Box Office wants to sell tickets for all these matches.

Each ticket transaction will have different ticketing-staff and customers involved with all critical information per customer, ticketing-staff, and transaction recorded.

Similarly, each match will also have other staff allotted: security-staff and café-staff.

All the three types of staff members (ticketing-staff, security-staff, café-staff) will have schedules allotted.

Your design could add more things to the existing requirements, but all the given requirements should be first met.

1. [3] Design the database that makes sense for the problem and select fields that make the most sense. A complete screenshot of your final design model is required.
2. [10] Determine the tables, columns, primary keys, nullabilities and show relationships between tables (one -one/one-many/many-many).
3. [5] Normalize your design into 3<sup>rd</sup> Normal Form. Please use MS Visio or any similar database design tool.
4. [2] Explain your design, including relationship between tables.

In your typed report (pdf or docx only), please include the following items:

1. Complete SQL source codes.
2. Comments for SQL codes (or for Screenshots, if codes aren't to be used for a particular question)
3. Full screenshots of the requested actions or each question including the total number of rows in result set and your name.
4. Your remarks on the project [4 pts.]. The remarks should be specific thoughts and references you learned that are related with knowledge points.

Submit your report on Blackboard.