**Part B: Case Study Exercises**

1. Write a SELECT statement that returns these column names and data from the OrderItems table:

ItemID                           The ItemID column

ItemPrice                      The ItemPrice column

DiscountAmount           The DiscountAmount column

Quantity                        The Quantity column

PriceTotal                     A column that’s calculated by multiplying the item price with the quantity

DiscountTotal                A column that’s calculated by multiplying the discount amount with the quantity

ItemTotal                       A column that’s calculated by subtracting the discount amount from the item price and then multiplying by the quantity

Only return rows where the ItemTotal is greater than 500.

Sort the result set by item total in descending sequence.

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

OrderID                         The OrderID column

OrderDate                     The OrderDate column

ShipDate                      The ShipDate column

Return only the rows where the ShipDate column contains a null value.

1. Write a SELECT statement without a FROM clause that creates a row with these columns:

Price                             100 (dollars)

TaxRate                        .07 (7 percent)

TaxAmount                   The price multiplied by the tax

Total                             The price plus tax

To calculate the fourth column, add the expressions you used for the first and third columns.

1. Write a SELECT statement that joins the Customers table to the Addresses table and returns these columns: FirstName, LastName, Line1, City, State, ZipCode.

Return one row for each address for the customer with an email address of [allan.sherwood@yahoo.com](mailto:allan.sherwood@yahoo.com).

1. Write a SELECT statement that joins the Customers, Orders, OrderItems, and Products tables. This statement should return these columns: LastName, FirstName, OrderDate, ProductName, ItemPrice, DiscountAmount, and Quantity.

Use aliases for the tables.

Sort the result set by LastName, OrderDate, and ProductName.

1. Use the UNION operator to generate a result set consisting of three columns from the Orders table:

ShipStatus              A calculated column that contains a value of SHIPPED or NOT SHIPPED

OrderID                   The OrderID column

OrderDate               The OrderDate column

If the order has a value in the ShipDate column, the ShipStatus column should contain a value of SHIPPED. Otherwise, it should contain a value of NOT SHIPPED.

Sort the final result set by OrderDate.

1. Write a SELECT statement that returns these columns:

The count of the number of orders in the Orders table

The sum of the TaxAmount columns in the Orders table

1. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:

The email address from the Customers table

The count of distinct products from the customer’s orders (Orders and OrderItems tables)

*Hint: Use two JOINS.*

1. 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.

1. Use a correlated subquery to return one row per customer, representing the customer’s oldest order (the one with the earliest date). Each row should include these three columns: EmailAddress, OrderID, and OrderDate.

*Hint: Use a JOIN and subquery*

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

ProductID:              The next automatically generated ID   
CategoryID:            4  
ProductCode:         dgx\_640  
ProductName:         Yamaha DGX 640 88-Key Digital Piano  
Description:            Long description to come.  
ListPrice:                799.99  
DiscountPercent:     0  
DateAdded:            Today’s date/time.

Use a column list for this statement.

1. Write an UPDATE statement that modifies the product you added in the previous exercise. This statement should change the DiscountPercent column from 0% to 35%.

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

The ListPrice column

A column that uses the CAST function to return the ListPrice column with one 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 ListPrice column as an integer

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

The OrderDate column

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

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

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

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

The OrderID column

The OrderDate column

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

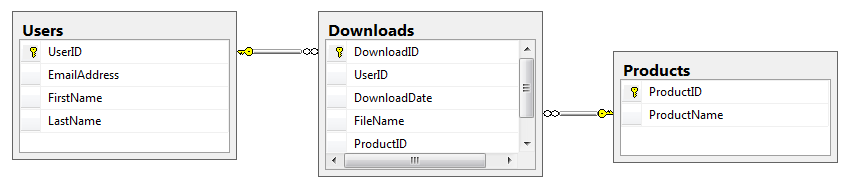
The ShipDate column

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 March 2012.

1. Create a database diagram that shows the relationships between the seven tables in the MyGuitarShop database. (The administrators table is not related to the other six tables.)

1. Use the Management Studio to create a new database called MyWebDB using the default settings. (If the database already exists, use the Management Studio to delete it and then recreate it.)
2. Use the Management Studio to create the following tables and relationships in MyWebDB.



1. Create a view named OrderItemProducts that returns columns from the Orders, OrderItems, and Products tables.

This view should return these columns from the Orders table: OrderID, OrderDate, TaxAmount, and ShipDate.

This view should return these columns from the OrderItems table: ItemPrice, DiscountAmount, FinalPrice (the discount amount subtracted from the item price), Quantity and ItemTotal (the calculated total for the item).

This view should return the ProductName column from the Products table.

*Hint:  Uses two JOINs.*

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 attempts to insert a new category named “Guitars” into the Categories table. If the insert is successful, the script should display this message:

SUCCESS: Record was inserted.

If the update is unsuccessful, the script should display a message something like this:

FAILURE: Record was not inserted.

Error 2627: Violation of UNIQUE KEY constraint 'UQ\_\_Categori\_\_8517B2E0A87CE853'. Cannot insert duplicate key in object 'dbo.Categories'. The duplicate key value is (Guitars).

*Hint: Use TRY and CATCH*