**Lab 4 Instructions**

1. Download the MyGuitarShop.sql file.
2. Modify the script so that the name of the database has your username before the MyGuitarShop database name.

IF DB\_ID('toddbreedloveMyGuitarShop') IS NOT NULL

DROP DATABASE toddbreedloveMyGuitarShop;

GO

CREATE DATABASE toddbreedloveMyGuitarShop;

GO

USE toddbreedloveMyGuitarShop;

1. Run the script to create the database.
2. Write the following SQL statements.
3. Check off as normal.

# Chapter 3

How to retrieve data from a single table

Exercises

Enter and run your own SELECT statements

In these exercises, you’ll enter and run your own SELECT statements.

1. Write a SELECT statement that returns four columns from the Products table: ProductCode, ProductName, ListPrice, and DiscountPercent. Then, run this statement to make sure it works correctly.

Add an ORDER BY clause to this statement that sorts the result set by list price in descending sequence. Then, run this statement again to make sure it works correctly. This is a good way to build and test a statement, one clause at a time.

1. Write a SELECT statement that returns one column from the Customers table named FullName that joins the LastName and FirstName columns.

Format this column with the last name, a comma, a space, and the first name like this:

Doe, John

Sort the result set by last name in ascending sequence.

Return only the contacts whose last name begins with a letter from M to Z.

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

ProductName The ProductName column

ListPrice The ListPrice column

DateAdded The DateAdded column

Return only the rows with a list price that’s greater than 500 and less than 2000.

Sort the result set in descending sequence by the DateAdded column.

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

ProductName The ProductName column

ListPrice The ListPrice column

DiscountPercent The DiscountPercent column

DiscountAmount A column that’s calculated from the previous two columns

DiscountPrice A column that’s calculated from the previous three columns

Sort the result set by discount price in descending sequence.

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 by the quantity

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

Work with nulls and test expressions

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 rate

Total The price plus tax

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

Chapter 4

How to retrieve data   
from two or more tables

Exercises

1. Write a SELECT statement that joins the Categories table to the Products table and returns these columns: CategoryName, ProductName, ListPrice.

Sort the result set by CategoryName and then by ProductName in ascending order.

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.

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 customer, but only return addresses that are the shipping address for a customer.

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 final result set by LastName, OrderDate, and ProductName.

1. Write a SELECT statement that returns the ProductName and ListPrice columns from the Products table.

Return one row for each product that has the same list price as another product. *(Hint: Use a self-join to check that the ProductID columns aren’t equal but the ListPrice column is equal.)*

Sort the result set by ProductName.

1. Write a SELECT statement that returns these two columns:

CategoryName The CategoryName column from the Categories table

ProductID The ProductID column from the Products table

Return one row for each category that has never been used. *(Hint: Use an outer join and only return rows where the ProductID column contains a null value.)*

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.

Chapter 5

How to code summary queries

Exercises

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 returns one row for each category that has products with these columns:

The CategoryName column from the Categories table

The count of the products in the Products table

The list price of the most expensive product in the Products table

Sort the result set so the category with the most products appears first.

1. Write a SELECT statement that returns one row for each customer that has orders with these columns:

The EmailAddress column from the Customers table

The sum of the item price in the OrderItems table multiplied by the quantiy in the OrderItems table

The sum of the discount amount column in the OrderItems table multiplied by the quantiy in the OrderItems table

Sort the result set in descending sequence by the item price total for each customer.

1. Write a SELECT statement that returns one row for each customer that has orders with these columns:

The EmailAddress column from the Customers table

A count of the number of orders

The total amount for those orders (*Hint: First, subtract the discount amount from the price. Then, multiply by the quantity.*)

Return only those rows where the customer has more than than 1 order.

Sort the result set in descending sequence by the sum of the line item amounts.

1. Modify the solution to exercise 4 so it only counts and totals line items that have an ItemPrice value that’s greater than 400.
2. Write a SELECT statement that answers this question: What is the total amount ordered for each product? Return these columns:

The product name from the Products table

The total amount for each product in the OrderItems table (*Hint: You can calculate the total amount by subtracting the discount amount from the item price and then multiplying it by the quantity*)

Use the WITH ROLLUP operator to include a row that gives the grand total.

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

Chapter 6

How to code subqueries

Exercises

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

1. Write a SELECT statement that answers this question: Which products have a list price that’s greater than the average list price for all products?

Return the ProductName and ListPrice columns for each product.

Sort the results by the ListPrice column in descending sequence.

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. Write a SELECT statement that returns three columns: EmailAddress, OrderID, and the order total 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 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.

1. Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don’t include products that have the same discount percent as another product.

Sort the results by the ProductName column.

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.

Chapter 7

How to insert, update, and delete data

Exercises

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.

1. Write an UPDATE statement that modifies the row you just added to the Categories table. This statement should change the CategoryName column to “Woodwinds”, and it should use the CategoryID column to identify the row.
2. 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.
3. 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 exercise 4. This statement should change the DiscountPercent column from 0% to 35%.
2. Write a DELETE statement that deletes the row in the Categories table that has an ID of 4. When you execute this statement, it will produce an error since the category has related rows in the Products table. To fix that, precede the DELETE statement with another DELETE statement that deletes all products in this category.
3. 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.

1. 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.
2. Write an UPDATE statement that modifies the Customers table. Change the password column to “reset” for every customer in the table.
3. 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

Exercises

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 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 ListPrice column as an integer

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

The DateAdded column

A column that uses the CAST function to return the DateAdded column with its date only (year, month, and day)

A column that uses the CAST function to return the DateAdded column with its full time only (hour, minutes, seconds, and milliseconds)

A column that uses the CAST function to return the DateAdded column with just the month and day

1. 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 2-digit months and days and a 4-digit year, 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 work with functions

Exercises

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

The ListPrice column

The DiscountPercent column

A column named DiscountAmount that uses the previous two columns to calculate the discount amount and uses the ROUND function to round the result to 2 decimal places

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’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.

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

The CardNumber column

The length of the CardNumber column

The last four digits of the CardNumber column

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.

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

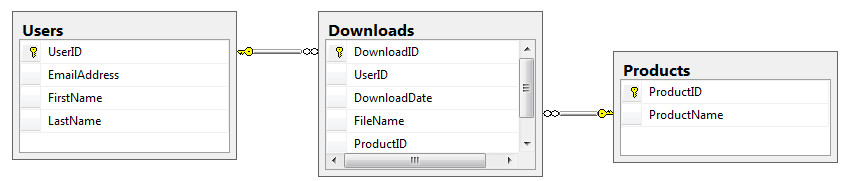
When you have this working, add a WHERE clause that retrieves just the orders for March 2016.

Chapter 11

How to create a database and its tables with SQL statements

Exercises

1. Write a script that adds an index to the MyGuitarShop database for the zip code field in the Addresses table.
2. Write a script that implements the following design in a database named MyWebDB:



In the Downloads table, the UserID and ProductID columns are the foreign keys.

Include a statement to drop the database if it already exists.

Include statements to create and select the database.

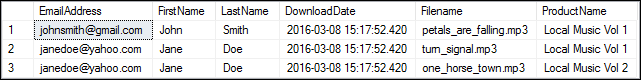
Include any indexes that you think are necessary.

1. Write a script that adds rows to the database that you created in exercise 2.

Add two rows to the Users and Products tables.

Add three rows to the Downloads table: one row for user 1 and product 2; one for user 2 and product 1; and one for user 2 and product 2. Use the GETDATE function to insert the current date and time into the DownloadDate column.

Write a SELECT statement that joins the three tables and retrieves the data from these tables like this:



Sort the results by the email address in descending order and the product name in ascending order.

1. Write an ALTER TABLE statement that adds two new columns to the Products table created in exercise 2.

Add one column for product price that provides for three digits to the left of the decimal point and two to the right. This column should have a default value of 9.99.

Add one column for the date and time that the product was added to the database.

1. Write an ALTER TABLE statement that modifies the Users table created in exercise 2 so the FirstName column cannot store null values and can store a maximum of 20 characters.

Code an UPDATE statement that attempts to insert a null value into this column. It should fail due to the not null constraint.

Code another UPDATE statement that attempts to insert a first name that’s longer than 20 characters. It should fail due to the length of the column.

Chapter 13

How to work with views

Exercises

1. Create a view named CustomerAddresses that shows the shipping and billing addresses for each customer in the MyGuitarShop database.

This view should return these columns from the Customers table: CustomerID, EmailAddress, LastName and FirstName.

This view should return these columns from the Addresses table: BillLine1, BillLine2, BillCity, BillState, BillZip, ShipLine1, ShipLine2, ShipCity, ShipState, and ShipZip.

Use the BillingAddressID and ShippingAddressID columns in the Customers table to determine which addresses are billing addresses and which are shipping addresses.

*Hint: You can use two JOIN clauses to join the Addresses table to Customers table twice (once for each type of address).*

1. Write a SELECT statement that returns these columns from the CustomerAddresses view that you created in exercise 1: CustomerID, LastName, FirstName, BillLine1.
2. Write an UPDATE statement that updates the CustomerAddresses view you created in exercise 1 so it sets the first line of the shipping address to “1990 Westwood Blvd.” for the customer with an ID of 8.
3. 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.

1. Create a view named ProductSummary that uses the view you created in exercise 4. This view should return some summary information about each product.

Each row should include these columns: ProductName, OrderCount (the number of times the product has been ordered), and OrderTotal (the total sales for the product).

1. Write a SELECT statement that uses the view that you created in exercise 5 to get total sales for the five best selling products.