# Lab 2: Basic SELECTs, Scripts, and Stored Procedures

**Objective**
After completing this lab, you will be able to:

- Use system stored procedures to retrieve information about the database.
- Write basic SELECT statements that return ordered and limited result sets.
- Modify and execute a query script.

You will be performing a few simple queries on your copy of the Northwind database to retrieve data out of a single table at a time. You will also correct and execute a simple SQL query script.

**Required Materials**
Laptop, Personal Northwind Database

**Related Reading**
SQL Server Books Online (Transact-SQL): sp_help, EXEC, SELECT, ORDER BY, ASC, DESC, AS, Round, ROWCOUNT.

**Turn-in Instructions**
For each of the tasks, provide (1) the SQL query you used along with (2) the results that you saved. Create a single .zip or .rar archive containing all this information and turn in the archive through ANGEL. (If you upload individual files we will not be able to download them and you will receive zero credit.)

**Assignment Details**
The Northwind database stores business information for Northwind Traders, a trading firm. Your task is to extract data from this database using SQL queries. Before you can do this, you need to learn a bit about how to use the query tools.

*Task 1: Getting Started with SQL Queries*
Open SQL Server Management Studio, then click the New Query button at the top-left of the screen.

*Task 2: Writing and Executing Basic Queries and Stored Procedures*
While many common database operations can be completed using Management Studio's built in wizards and prompts, relational databases convert GUI actions to code and execute that dynamic code. In SQL Server, this language is SQL or Structured Query Language. Over the next several weeks you will be writing a variety of queries, or scripts, that will execute on your database system. Although you can write your queries in any text editor, SQL Server Management Studio provides an excellent environment in which you can test and execute your queries without having to load external applications. If at any time you are having difficulty with a SQL command, first try looking it up in the SQL Index available from the Help menu in SQL Server Management Studio. This knowledge base has usage diagrams and sample queries for each of SQL's keywords.

SQL has a few built-in stored procedures that you can run to get more information about

the database you are working with. The first one you will experiment with will be **sp_help**.

In the Query window type: `sp_help` then execute the SQL query by clicking on the red exclamation point, selecting Query > Execute, or by pushing the F5 key. You will see a number of things occurring on screen, at the bottom of the Query window are a number of status indicators, including the Query Status, Server Name, Username, Database Name, Query Run-time, and the number of rows returned by the Query. This area will be useful when working directly with raw SQL queries or when needing to see how big a query is. After a few seconds, the Query window will change and you will be shown information about the database server. Scroll through the list to see what kinds of information is returned by the raw sp_help stored procedure.

Writing a single instruction in a query would grow tiresome quickly, so SQL is capable of handling multiple operations in one .sql file. Before proceeding with the next SQL query, take a moment to start a new query tab by clicking on the "New Query" button in the upper-left corner of the screen. Type the following lines of code into the code editor window substituting the name of your personal Northwind database in for NWindUsernameNN, then execute the query:

```
USE NWindUsernameNN
EXEC sp_help
```

As before, the query will take a few seconds to complete, and the window will refresh. Unlike the last query, the screen now displays information specific to your copy of the Northwinds Database. Examine this result set. Using the `USE <DB Name>` command restricts sp_help to showing information about the current database instead of the entire database server. You can also specify sp_help on specific items within your database. Using the last query as your base, add the argument 'Orders' to your query, then execute it. Again, if you are unsure about how to supply arguments, lookup sp_help in the Index of the Help menu.

Yet again, the data that is returned from your query is different, take a moment to examine the differences between it and the last query. Later in this lab, you may need the results of sp_help to figure out how to construct more interesting queries.

You can also save queries with Management Studio. Open a "New Query", key in the following query, then execute it. Remember to substitute your Database name where appropriate.

```
USE NWindUsernameNN
SELECT ProductName, UnitsInStock
     FROM Products WHERE UnitsOnOrder > 0
```

Examine the output and the SQL query. The output should include 17 rows. As you see, the `SELECT` statement has only returned records (rows) that match the column names in the Products Table that have one or more items on order. You will be using similar syntax in the next several tasks.

Navigate through Management Studio for a save option and save the above query to your computer, naming it "Task2.sql". You should also save the query result as "Task2Results.csv". (Try right-clicking the results area.) In this and subsequent labs you will be required to save and submit your SQL queries to ANGEL for grading.

*Task 3: Writing SELECT Statements from a Result Set*
Imagine that your lazy colleague Wally gives you a hard copy of the results of an SQL SELECT that looks like this:

| ***ProductID*** | ***ProductName*** | ***UnitsInStock*** | ***UnitPrice*** |
|---|---|---|---|
| 60 | Camembert Pierrot | 19 | 34.0000 |
| 2 | Chang | 17 | 19.0000 |
| 38 | Côte de Blaye | 17 | 263.5000 |
| 43 | Ipoh Coffee | 17 | 46.0000 |
| 62 | Tarte au sucre | 17 | 49.3000 |
| 48 | Chocolade | 15 | 12.7500 |
| 26 | Gumbär Gummibärchen | 15 | 31.2300 |

*Figure 1: The top 7 of 26 records in Wally's Hardcopy*

Wally tells you that this query lists all the products with fewer than 20 items in stock. Unfortunately, Wally has lost the original query that generated this data readout, and updated data from this readout is now needed. Because Wally is busy drinking coffee, your task is to write a query on the Products table that will generate a data readout with the same order characteristics as Wally's hardcopy. You may find the `ORDER BY` clause helpful in sorting the data returned by your query as well as the Object Explorer (left pane) for finding Table and Column names.

After completing the task, save your query and the query results to the same folder as your answers from Task 2, but naming them for Task 3.

*Task 4: Creating a Calculating Column*
Next, you need to find actual unit prices (after discounts, rounded to the nearest dollar) for all sales of at least 100 units of product. Using the Order Details table, write an SQL query that will return ProductID, Quantity, UnitPrice, Discount, and the actual cost (including discounts) rounded to the nearest dollar. Note that discounts are listed as percent off; that is, 0.25 means a 25% discount. All columns in your result should have descriptive headings, *i.e., not* "(No column name)". This naming should be done within the SQL query, not by externally manipulating the data. Sort your results in ascending order by quantity then by product ID.

After completing this task, save your query and the query results with your previous files, using the same naming convention.

*Task 5: Debugging SQL Queries*
Frequently, you will have a query or stored procedure that is not returning values you expect, or does not run at all. Copy the following SQL Query into a New Query page, the read the English description that follows. Make corrections to the SQL query so that it reflects the English description of that query.

```
SET ROWCOUNT 6;
SELECT ProductName,QuantityPerUnit,UnitPrice
     FROM Products
     ORDER BY UnitPrice ASC
     WHERE CategoryID = 4;
```

This script is intended to return the product name, quantity per unit, and unit price of the five highest-priced category 4 products. It contains both logic and syntax errors. Enter the script in a new query. Fix the script so that it returns the intended result, using Management Studio help as appropriate.

After completing this task, save your query and the query results with your previous files.

**Turn-in Instructions**
For **each** of the tasks above, provide (1) the SQL query you used along with (2) the results that you saved. Create a single .zip or .rar archive containing all this information and turn in the archive through ANGEL. (If you upload individual files we will not be able to download them and **you will receive zero credit**.)

Please complete the anonymous lab feedback survey on Angel under Materials -> Lab Feedback. Your feedback will help us improve the labs for future students.

**Revision History**

| | |
|---|---|
| Dec. 2, 2004: | Initial revision by David Yip. |
| Dec. 3, 2004: | Revisions made by Patrick Roby. |
| Dec. 6, 2004: | Additional objectives added by David Yip. |
| Dec. 8, 2004: | Clarification of objectives done by David Yip. |
| Nov. 16, 2005: | Cleaned up tasks, dropped osql. Curt Clifton and Steve Chenoweth |
| Oct. 23, 2006: | Updated for SQL Server 2005 by Andrew Foltz |
| Oct. 31, 2006: | Updated and revised Tasks by Bryan Musial |
| Dec. 6, 2006: | Minor tweaks and clarifications by Curt Clifton |