MICROSOFT
TRAINING
AND CERTIFICATION

Microsoft Official Curriculum

Module 2: Using Transact-SQL Querying Tools

Contents

Microsoft

Project Lead: Cheryl Hoople
Instructional Designer: Cheryl Hoople
Technical Lead: LeRoy Tuttle
Program Manager: LeRoy Tuttle
Graphic Artist: Kimberly Jackson (Independent Contractor)
Editing Manager: Lynette Skinner
Editor: Wendy Cleary
Editorial Contributor: Elizabeth Reese
Copy Editor: Bill Jones (S&T Consulting)
Production Manager: Miracle Davis
Production Coordinator: Jenny Boe
Production Tools Specialist: Julie Challenger
Production Support: Lori Walker (S&T Consulting)
Test Manager: Sid Benavente
Courseware Testing: Testing Testing 123
Classroom Automation: Lorrin Smith-Bates
Creative Director, Media/Sim Services: David Mahlmann
Web Development Lead: Lisa Pease
CD Build Specialist: Julie Challenger
Online Support: David Myka (S&T Consulting)
Localization Manager: Rick Terek
Operations Coordinator: John Williams
Manufacturing Support: Laura King; Kathy Hershey
Lead Product Manager, Release Management: Bo Galford
Lead Product Manager: Margo Crandall
Group Manager, Courseware Infrastructure: David Bramble
Group Product Manager, Content Development: Dean Murray
General Manager: Robert Stewart

Instructor Notes

Presentation:
30 Minutes

Lab:
30 Minutes

Microsoft® SQL Server™ 2000 provides several querying tools that you can use to execute Transact-SQL scripts. This module describes how to use SQL Query Analyzer and the **osql** command-line utility and how to execute Transact-SQL statements in various ways.

At the end of this module, you will be able to:

- Describe the basic functions of SQL Query Analyzer.
- Describe how to use the Object Browser tool in SQL Query Analyzer.
- Describe how to use the templates in SQL Query Analyzer.
- Describe how to use the **osql** command-line utility.
- Execute Transact-SQL statements in various ways.

Materials and Preparation

This section provides you with the required materials and preparation tasks that are needed to teach this module.

Required Materials

To teach this module, you need the following materials:

- Microsoft PowerPoint® file 2071A_02.ppt.
- The C:\MOC\2071A\Demo\Ex_02.sql example file, which contains all of the example scripts from the module, unless otherwise noted in the module.

Preparation Tasks

To prepare for this module, you should:

- Read all of the materials for this module.
- Complete the lab.

Module Strategy

Use the following strategy to present this module:

- SQL Server Query Analyzer

  Introduce SQL Query Analyzer. Demonstrate the basic functions of SQL Query Analyzer, pointing out that students can execute part or all of a query, execute it into a grid, and create an execution plan. Point out that SQL Query Analyzer automatically color-codes the syntax, that students can have multiple query windows, and that students can execute parts of the script.

- Using the Object Browser Tool in SQL Query Analyzer

  Emphasize that students can use the Object Browser tool in SQL Query Analyzer to locate and script objects and to eliminate many typing and syntax errors.

- Using Templates in SQL Query Analyzer

  Describe the purpose and use of the templates that SQL Query Analyzer provides. Briefly demonstrate how to gain access to these templates and show how the graphical interface works. Review the template parameter definitions.

- Using the **osql** Command-line Utility

  Describe when and how to use the **osql** command-line utility. If students ask about the **isql** utility, point out that it is not included in this course because it uses DB-Library to communicate with the server and does not support Unicode data types.

- Executing Transact-SQL Statements

  Familiarize students with the various ways that they can execute Transact-SQL statements. These include dynamically constructing statements, submitting batches, and running scripts. Where possible, demonstrate these by using SQL Query Analyzer.

Customization Information

This section identifies the lab setup requirements for a module and the configuration changes that occur on student computers during the labs. This information is provided to assist you in replicating or customizing Microsoft Official Curriculum (MOC) courseware.

Important The lab in this module is dependent on the classroom configuration that is specified in the Customization Information section at the end of the *Classroom Setup Guide* for course 2071A, *Querying Microsoft SQL Server 2000 with Transact-SQL*.

Lab Setup

There are no lab setup requirements that affect replication or customization.

Lab Results

There are no configuration changes on student computers that affect replication or customization.

Overview

Slide Objective
To provide an overview of the module topics and objectives.

Lead-in
In this module, you will learn about some of the querying tools that SQL Server provides for executing Transact-SQL scripts.

- **SQL Query Analyzer**
- **Using the Object Browser Tool in SQL Query Analyzer**
- **Using Templates in SQL Query Analyzer**
- **Using the osql Utility**
- **Executing Transact-SQL Statements**

Microsoft® SQL Server™ 2000 provides several querying tools that you can use to execute Transact-SQL scripts. This module describes how to use SQL Query Analyzer and the **osql** command-line utility and how to execute Transact-SQL statements in a number of different ways.

At the end of this module, you will be able to:

- Describe the basic functions of SQL Query Analyzer.
- Describe how to use the Object Browser tool in SQL Query Analyzer.
- Describe how to use the templates in SQL Query Analyzer.
- Describe how to use the **osql** command-line utility.
- Execute Transact-SQL statements in a various ways.

SQL Query Analyzer

Slide Objective
To introduce the SQL Query Analyzer tool.

Lead-in
You use SQL Query Analyzer in SQL Server to view query statements and results graphically.

- **Free-Form Text Editor**
- **Color-Coding of Transact-SQL Syntax**
- **Multiple Query Windows with Grid or Text Output**
- **Partial Script Execution**
- **Query Execution Information**

Delivery Tip
Demonstrate SQL Query Analyzer. Show students the basic elements of the SQL Query Analyzer window, including the three ways to execute a query, the syntax color-coding, and opening and saving a script.

You use SQL Query Analyzer in SQL Server to view query statements and results graphically. You also can use it for writing, modifying, and saving Transact-SQL scripts.

SQL Query Analyzer also provides tools for determining how SQL Server is interpreting and working with a Transact-SQL statement.

SQL Query Analyzer includes:

- *A free-form text editor.* This editor has advanced text-editing capabilities such as block indents, block comment or un-comment, and conversion to upper- or lower-case.
- *Color-coding.* As you write a query, SQL Query Analyzer highlights keywords, character strings, and other language elements, and you can customize how they appear by using color-coding.
- *Multiple query windows with grid or text output.* Each query window has its own connection to a SQL Server. You can view results in a text window or in a grid.
- *Partial script execution.* This capability allows you to execute portions of a script. When you can select portions of a script, SQL Server executes only those portions.
- *Query execution information.* Query execution information includes such things as client statistics, server trace information, and execution plan data. You can use this information to help tune and troubleshoot your scripts.

Using the Object Browser Tool in SQL Query Analyzer

Slide Objective
To introduce the Object Browser tool within SQL Query Analyzer.

Lead-in
You can use the Object Browser tool within SQL Query Analyzer to navigate the tree view of the objects in a database and drill down to a specific object.

- **The Object Browser Enables Navigation of the Tree View of Objects in a Database**
- **Using the Object Browser, You Can:**
  - Script objects
  - Execute stored procedures
  - Open tables
  - Alter objects in the database
  - Use Transact-SQL templates

You can use the Object Browser tool within SQL Query Analyzer to navigate the tree view of objects in a database and drill down to a specific object. The Object Browser also scripts objects, executes stored procedures, and allows you to gain access to tables and views.

Using the Object Browser, you can:

- Script objects.

  The operations that the Object Browser supports vary, depending on the type of object. For example, table objects can generate scripts containing SELECT statements, data definition statements such as CREATE, or data manipulation statements such as INSERT.

- Execute stored procedures.

  When you execute a stored procedure that has a parameter, the Object Browser prompts for values.

- Open tables.

  The Object Browser displays query results separately. You can edit, insert, or delete rows.

- Alter objects in the database.

  You can view and edit objects in a database. The Object Browser displays an ALTER statement for the selected object in the Editor pane. For example, if the selected object is a stored procedure, the Object Browser provides an ALTER PROCEDURE statement. You can use this ALTER statement to specify the changes, and then execute it.

- Use Transact-SQL templates.

  These templates contain Transact-SQL scripts that help you create objects in the database. You can use these templates to:

  - Create databases, tables, views, indexes, stored procedures, triggers, statistics, and functions.
  - Manage extended properties, linked servers, logon accounts, roles, and users.
  - Declare and use cursors.
  - Customize scripts.

Using Templates in SQL Query Analyzer

Slide Objective
To describe how to use templates in SQL Query Analyzer.

Lead-in
SQL Query Analyzer includes templates that you can use as starting points for creating objects in a database.

- **Templates**
  - Are starting points for creating objects in a database
  - Contain parameters to help you customize scripts
- **Format for Template Parameter Definitions**

  <parameter_name, data_type, value>

Delivery Tip
Describe the purpose and use of the templates that SQL Query Analyzer provides.

Briefly demonstrate how to gain access to these templates and show how the graphical interface works.

Refer students to SQL Server Books Online for more information.

SQL Query Analyzer includes templates that you can use as starting points for creating objects in a database.

SQL Server provides a variety of templates in the Templates\SQL Query Analyzer directory. Among the templates provided are those that create databases, tables, views, indexes, stored procedures, triggers, statistics, and functions. Other templates in this directory help you manage extended properties, linked servers, logins, roles, and users, and help you to declare and use cursors.

The template scripts provided with SQL Query Analyzer contain parameters to help you customize scripts. Template parameter definitions use this format:

```
<parameter_name, data_type, value>
```

The following table describes the format and template parameter definitions:

Format	Template parameter definition
<parameter_name>	Name of the parameter in the script
<data_type>	Data type of the parameter
<value>	Value that is to replace every occurrence of the parameter in the script

You use a dialog box to insert values into the script. For example, when you execute a function from Object Browser, the function that is written to the Edit pane contains parameter definitions for any arguments in the function. You then use the Replace Template Parameters dialog box to specify argument values.

Using the osql Utility

Slide Objective
To introduce the **osql** utility.

Lead-in
The **osql** utility is a command line utility for querying SQL Server.

- **Starting the osql Command-line Utility**
- **Using the Interactive Mode**
- **Using the Script Execution Mode**
- **Using Extended osql Scripting Commands**

For Your Information
The **isql** utility is not covered in this course because it uses DB-Library to communicate with the server and does not support Unicode data types.

The ***osql** utility* is a command-line utility for ad hoc, interactive execution of Transact-SQL statements and scripts. To use the **osql** command-line utility, users must understand Transact-SQL and know how to execute scripts from a command prompt.

The **osql** command-line utility uses SQL Server Open Database Connectivity (ODBC) to communicate with the server and is subject to the restrictions and behaviors of the ODBC application programming interface (API).

Starting the osql Command-line Utility

You start the **osql** command-line utility directly from the operating system with the case-sensitive options listed below. You can call it from a batch file or from a command prompt. A *batch* is a set of Transact-SQL statements that are submitted together and executed as a group.

Using the Interactive Mode

The **osql** command-line utility accepts Transact-SQL statements and sends them to SQL Server interactively. The results are formatted and displayed on the screen.

Use the GO statement to execute Transact-SQL statements in the input buffer. Use the QUIT or EXIT statement to exit the **osql** command-line utility.

Using the Script Execution Mode

Users submit an **osql** batch specifying a single Transact-SQL statement to execute or pointing the utility to a text file that contains Transact-SQL statements to execute. The output is usually directed to a text file, but the output also can be displayed in the command prompt window.

Partial Syntax

osql [**-S** *server_name*] [**-E**] [**-U** *login_id*] [**-P** *password*]
[**-i** *input_file*] [**-o** *output_file*] [**-?**]

Note Parameters in **osql** statements are case sensitive.

Delivery Tip
Mention that the dash (-) or forward slash (/) character can precede arguments.

For more information on arguments, refer students to SQL Server Books Online, search topic "osql utility".

Remember that the dash (-) or forward slash (/) character can precede arguments. The following table describes the most commonly used arguments.

Argument	Description
-S *server_name*	Specifies the SQL Server to which to connect. The *server_name* is the name of the server computer on the network. This option is required if you execute **osql** from a remote computer on the network.
-E	Uses a trusted connection instead of requesting a password.
-U *login_id*	Is the user login ID. Login IDs are case sensitive. If neither the **-U** or **-P** option is used, SQL Server uses the currently logged in user account and will not prompt for a password.
-P *password*	Is a user-specified password. If the **-P** option is not used, **osql** prompts for a password. If the **-P** option is used at the end of the command prompt without any password, **osql** uses the default password (NULL). Passwords are case sensitive. If neither the **-U** or **-P** option is used, SQL Server uses the currently logged in user account and will not prompt for a password.
-i *input_file*	Identifies the file that contains a batch of Transact-SQL statements or stored procedures. The less than (<) symbol can be used in place of **-i**.
-o *output_file*	Identifies the file that receives output from **osql**. The greater than (>) symbol can be used in place of **-o**. If the input file is Unicode, the output file will be Unicode if **-o** is specified. If the input file is not Unicode, the output file is OEM.
-?	Displays the syntax summary of **osql** switches.

Using Extended osql Scripting Commands

The **osql** command-line utility can also process commands that are not Transact-SQL statements. The osql command-line utility only recognizes these commands when they occur at the beginning of a line or immediately following the **osql** prompt. It disregards all subsequent statements on the same line.

The following table describes these additional commands.

Command	Description
GO	Executes all statements entered after the last GO
RESET	Clears any statements that you have entered
ED	Calls the editor
!! *command*	Executes an operating-system command
QUIT or EXIT( )	Exits from **osql**
CTRL+C	Ends a query without exiting from **osql**

Executing Transact-SQL Statements

Slide Objective
To provide an overview of the ways to execute Transact-SQL statements.

Lead-in
You can execute Transact-SQL statements in a variety of ways.

- **Dynamically Constructing Statements**
- **Using Batches**
- **Using Scripts**

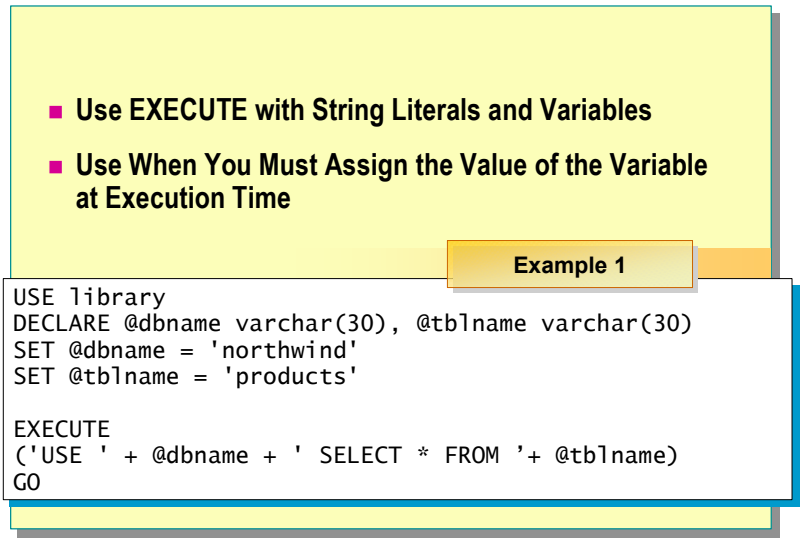
You can execute Transact-SQL statements in a variety of ways by:

- Dynamically constructing statements at run-time.
- Using batches to group statements that should be run together.
- Using scripts to save batches to a file for later use.

Dynamically Constructing Statements

Slide Objective
To introduce students to the dynamic execution of statements.

Lead-in
You can build statements dynamically so that they are constructed at the same time that a script is executed.



You can build statements dynamically so that they are constructed at the same time that a script is executed.

To build a statement dynamically, use the EXECUTE statement with a series of string literals and variables that are resolved at execution time.

Dynamically constructed statements are useful when you want the value of the variable to be assigned when the statement executes. For example, you can create a dynamic statement that performs the same action on a series of database objects.

Syntax

EXECUTE **(**{*@str_var* | *'tsql_string'*} + [{*@str_var* | *'tsql_string'*}...]**)**}

Consider the following facts about the EXECUTE statement:

- The EXECUTE statement executes statements composed of character strings within a Transact-SQL batch. Because these are string literals, be sure that you add spaces in the appropriate places in order to ensure proper concatenation.
- The EXECUTE statement can include a string literal, a string local variable, or a concatenation of both.
- All items within the EXECUTE string must consist of character data; you must convert all numeric data before you use the EXECUTE statement.
- You cannot use functions to build the string for execution in the EXECUTE statement.
- You can create any valid Transact-SQL statements dynamically, including functions.
- You can nest EXECUTE statements.
- Variables and temporary tables that are created dynamically last only as long as it takes for the statement to execute.

Example 1

This example demonstrates how a dynamically executed statement is used to specify a database context different than that in which you are currently and then to select all the columns and rows from a specified table. In this example, the change of the database context to the **northwind** database lasts only for the duration of the query. The current database context is unchanged.

Using a stored procedure, the user could pass the database and table information into the statement as parameters and then query a specific table within a database.

```
USE library
DECLARE @dbname varchar(30), @tablename varchar(30)
SET @dbname = 'northwind'
SET @tablename = 'products'

EXECUTE
  ('USE ' + @dbname + ' SELECT productname FROM ' +
@tablename)
GO
```

Result

```
productname
Chai
Chang
Aniseed Syrup
```

Example 2

This example demonstrates how a dynamically executed statement can be used to change a database option for the duration of the statement. The following statement does not return a count of the number of rows affected.

```
USE northwind
EXECUTE ('SET NOCOUNT ON '+ 'SELECT lastname, reportsto FROM
employees WHERE reportsto IS NULL')
GO
```

Result

```
lastname                  reportsto
Fuller                    NULL
```

Using Batches

Slide Objective
To introduce students to using batches.

Lead-in
A batch is a set of Transact-SQL statements that are submitted together and executed as a group.

- **One or More Transact-SQL Statements Submitted Together**
- **Defining a Batch with the GO Statement**
- **How SQL Server Processes Batches**
- **Statements That You Cannot Combine in a Batch**
  - CREATE PROCEDURE
  - CREATE VIEW
  - CREATE TRIGGER
  - CREATE RULE
  - CREATE DEFAULT

You can also submit one or more statements in a batch.

One or More Transact-SQL Statements Submitted Together

A *batch* is a set of Transact-SQL statements that are submitted together and executed as a group. Batches can be run interactively or as part of a script. A script can include more than one batch of Transact-SQL statements.

Defining a Batch with the GO Statement

Use a GO statement to signal the end of a batch. GO is not a universally accepted Transact-SQL statement; it is a statement accepted only by SQL Query Analyzer and the **osql** utility. Applications based on the ODBC or OLE DB APIs generate a syntax error if they attempt to execute a GO statement.

How SQL Server Processes Batches

SQL Server optimizes, compiles, and executes the statements in a batch together. However, the statements do not necessarily execute as a recoverable unit of work.

The scope of user-defined variables is limited to a batch, so a variable cannot be referenced after a GO statement.

Note If a syntax error exists in a batch, none of the statements in that batch executes. Execution begins with the next batch.

Statements That You Cannot Combine in a Batch

Delivery Tip
Describe and compare each example.

You must execute certain object creation statements in their own batches within a script, because of the way that the object creation statements are defined. These object creation statements are indicated by a pattern—an object definition header, followed by the AS keyword with one or more definition statements, and concluded by a GO command.

You must execute these statements in separate batches:

- CREATE PROCEDURE
- CREATE VIEW
- CREATE TRIGGER
- CREATE RULE AS
- CREATE DEFAULT

Example 1

This example shows statements would fail as executed as part of a single batch, because the query improperly combines statements that cannot be combined in a batch. You must insert a GO statement before each CREATE VIEW statement to execute this statement correctly.

```
CREATE DATABASE ...
CREATE TABLE ...
CREATE VIEW ...
CREATE VIEW ...
GO
```

Example 2

This example groups the statements used in Example 1 into proper batches so that they execute correctly.

```
CREATE DATABASE ...
CREATE TABLE ...
GO

CREATE VIEW ...
GO

CREATE VIEW ...
GO
```

Using Scripts

Slide Objective
To introduce using scripts to execute Transact-SQL statements.

Lead-in
Scripts are one of the most common ways to execute Transact-SQL statements.

- **A Script Is One or More Transact-SQL Statements Saved as a File Using the .sql Extension**
- **Scripts:**
  - Contain saved statements
  - Can be written using any text editor
  - Can recreate database objects or execute statements repeatedly
  - Execute in SQL Query Analyzer or in the **osql** utility

Scripts are one of the most common ways to execute Transact-SQL statements. They are one or more Transact-SQL statements that are saved as a file.

You can write and save scripts in SQL Query Analyzer or in any text editor such as Notepad. Save the script file with the .sql file name extension.

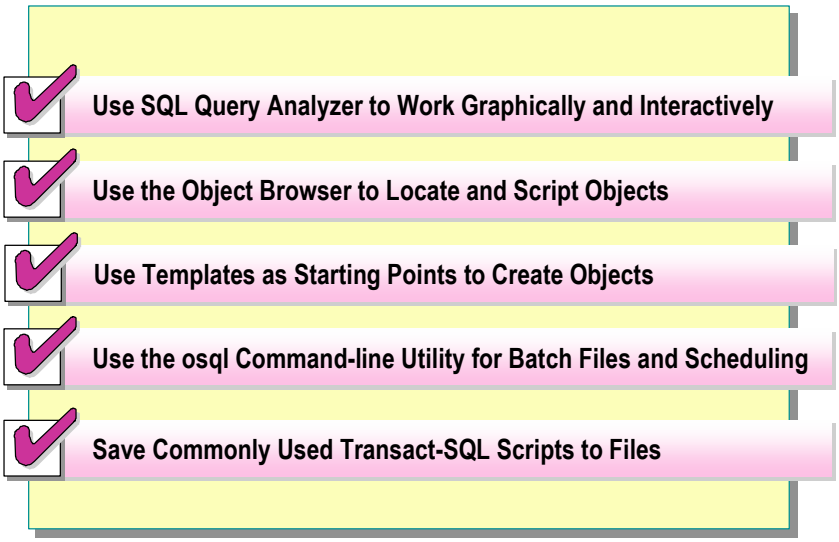
Saved scripts are very useful when you want to recreate databases or data objects, or when you must use a set of statements repeatedly.

You can open and execute the script file in SQL Query Analyzer or use the **osql** utility (or another query tool).

Recommended Practices

Slide Objective
To present recommended practices for using Transact-SQL querying tools.

Lead-in
These recommended practices should help you use Transact-SQL querying tools.



The following recommended practices should help you use Transact-SQL querying tools:

- Use SQL Query Analyzer when you want to work graphically and interactively. You can use multiple connections to SQL Server and cut and paste between windows, while leveraging the color-coding of syntax and the scripting ability of the Object Browser tool.
- Use the Object Browser tool to locate and script table and column names and to create error-free scripts that alter objects and modify data.
- Use templates in SQL Query Analyzer as starting points for creating objects in a database.
- Use the **osql** command-line utility for batch files and for the execution of repetitive tasks. Additional scripting features with the **osql** command-line utility environment can benefit automation and maintenance tasks.
- Save commonly used Transact-SQL scripts to files. These files effectively constitute a library of reusable scripts for consistency and future use.

Additional information on the following topics is available in SQL Server Books Online.

Topic	Search for
Using SQL Query Analyzer	"Overview of SQL Query Analyzer"
Using the **osql** utility	"osql utility"

Lab A: Creating and Executing Transact-SQL Scripts

Slide Objective
To introduce the lab.

Lead-in
In this lab, you will create a Transact-SQL script, save it, and then execute it in SQL Query Analyzer and through the **osql** utility.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Write basic SELECT statements that return ordered and limited result sets.
- Modify and execute a script by using the **osql** utility.

Prerequisites

Before working on this lab, you must have:

- Script files for this lab, which are located in C:\Moc\2071A\Labfiles\L02.
- Answer files for this lab, which are located in C:\Moc\2071A\Labfiles\L02\Answers.

For More Information

If you require help in executing files, search SQL Query Analyzer Help for "Execute a query".

Other resources that you can use include:

- The **Northwind** database schema.
- Microsoft® SQL Server™ Books Online.

Scenario

The organization of the classroom is meant to simulate that of a worldwide trading firm named Northwind Traders. Its fictitious domain name is nwtraders.msft. The primary DNS server for nwtraders.msft is the instructor computer, which has an Internet Protocol (IP) address of 192.168.*x*.200 (where *x* is the assigned classroom number). The name of the instructor computer is London.

The following table provides the user name, computer name, and IP address for each student computer in the fictitious nwtraders.msft domain. Find the user name for your computer, and make a note of it.

User name	Computer name	IP address
SQLAdmin1	Vancouver	192.168.x.1
SQLAdmin2	Denver	192.168.x.2
SQLAdmin3	Perth	192.168.x.3
SQLAdmin4	Brisbane	192.168.x.4
SQLAdmin5	Lisbon	192.168.x.5
SQLAdmin6	Bonn	192.168.x.6
SQLAdmin7	Lima	192.168.x.7
SQLAdmin8	Santiago	192.168.x.8
SQLAdmin9	Bangalore	192.168.x.9
SQLAdmin10	Singapore	192.168.x.10
SQLAdmin11	Casablanca	192.168.x.11
SQLAdmin12	Tunis	192.168.x.12
SQLAdmin13	Acapulco	192.168.x.13
SQLAdmin14	Miami	192.168.x.14
SQLAdmin15	Auckland	192.168.x.15
SQLAdmin16	Suva	192.168.x.16
SQLAdmin17	Stockholm	192.168.x.17
SQLAdmin18	Moscow	192.168.x.18
SQLAdmin19	Caracas	192.168.x.19
SQLAdmin20	Montevideo	192.168.x.20
SQLAdmin21	Manila	192.168.x.21
SQLAdmin22	Tokyo	192.168.x.22
SQLAdmin23	Khartoum	192.168.x.23
SQLAdmin24	Nairobi	192.168.x.24

Estimated time to complete this lab: 30 minutes

Exercise 1
Writing Basic SELECT Statements

In this exercise, you will write various statements that return rows from the **products** table in the **Northwind** database.
C:\Moc\2071A\Labfiles\L02\Answers contains completed scripts for this exercise.

► To write a SELECT statement that returns ordered data

In this procedure, you will write a statement that returns all the rows and columns from the **products** table and sorts the results in ascending order by the **productname** column. Answer_Ordered.sql is a completed script for this procedure.

1. Log on to the **NWTraders** classroom domain by using the information in the following table.

Option	Value
User name	**SQLAdmin*x*** (where *x* corresponds to your computer name as designated in the nwtraders.msft classroom domain)
Password	**password**

2. Open SQL Query Analyzer and, if requested, log in to the (local) server with Microsoft Windows® Authentication.

   You have permission to log in to and administer SQL Server because you are logged as **SQLAdmin*x***, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

3. In the **DB** list, click **northwind**.
4. Type and execute a SELECT statement that returns all the rows and columns from the **products** table and sorts the results in ascending order by the **productname** column.

   You can execute the **sp_help** system stored procedure on the **products** table to find the correct column names.

   ```
   SELECT * FROM products ORDER BY productname
   ```

5. Click the **Execute Mode to Results in Grid** toolbar button.
6. Execute the statement again.

► To write a SELECT statement that returns limited data

In this procedure, you will write a statement that retrieves products from a specific category. Answer_Limited.sql is a completed script for this procedure.

- Type and execute a SELECT statement that retrieves all products in category (**categoryid**) 4 from the **products** table.

  You can execute the **sp_help** system stored procedure on the **products** table to find the correct column names.

```
SELECT * FROM products WHERE categoryid = 4
GO
```

Tip To see more information about the SELECT statement (as well as any Transact-SQL statement and system table), select the **SELECT** keyword in the query window, right-click the **SELECT** keyword, and then click **Transact-SQL Help**. Double-click **SELECT: clauses**.

Exercise 2
Modifying a Script File

In this exercise, you will modify, save, and execute a simple script file. C:\Moc\2071A\Labfiles\L02\Answers contains completed scripts for this exercise.

► To modify a script file

In this procedure, you will execute a script that contains errors. By using the error information that is returned, you will make changes to the script so that it executes correctly. Then, you will save and execute the script.

1. Open C:\Moc\2071A\Labfiles\L02\MyScript.sql, review, and then execute it. You will get errors when you run this file. These errors are intentional. Answer_MyScript.sql is a completed script for this procedure.
2. Type block comment keywords around the lines of the script name and description.

```
/*
            MYSCRIPT.SQL

  This script queries the customer table and
  returns a list of customer numbers and their
  last names.
  This script should be run in the northwind database.
*/
```

3. Add a statement that specifies that the script will run in the context of the **Northwind** database.

```
USE northwind
```

4. Include end of batch markers (GO statements) in the proper areas of the script. Only two additional batch markers are needed.

```
SELECT cust_no, lname FROM sample1
SELECT CompanyName FROM customers
GO
```

5. Save the script and then execute it.
6. Minimize SQL Query Analyzer.

Exercise 3
Execute a Script Using the osql Utility

In this exercise, you will use the **osql** command-line utility to execute the script that you created in exercise 2.

► To display the osql command-line utility arguments

In this procedure, you will display the **osql** command-line utility arguments. Open a command prompt window.

1. Type the following command to display the **osql** command-line arguments.

   ```
   osql -?
   ```

2. Review the arguments.

► To execute a script file by using the osql utility

In this procedure, you will execute a script file by using the **osql** utility. The -E argument specifies that a trusted connection should be made to SQL Server.

1. Open a command prompt window.
2. Type the following command to execute C:\Moc\2071A\Labfiles\L02\MyScript.sql. Make sure that the path is correct.

   ```
   osql -E -i "C:\MOC\2071A\labfiles\L02\MyScript.sql"
   ```

   Note If the -S argument is not used to specify the SQL Server to which the **osql** utility connects, then the **osql** utility connects to the local SQL Server by default.

3. Verify that the results are the same as those obtained in exercise 2.

Review

Slide Objective
To reinforce module objectives by reviewing key points.

Lead-in
The review questions cover some of the key concepts taught in the module.

- SQL Server Query Analyzer
- Using the Object Browser Tool in SQL Query Analyzer
- Using Templates in SQL Query Analyzer
- Using the osql Utility
- Executing Transact-SQL Statements

1. What is the best querying tool to use within a batch file to capture the results of a query in a text file? Why?

   It is best to use the osql command-line utility to execute the query and save the results to a text file by using the command-line option–o *filename.txt* parameter.

   It is also possible to use appropriate command-line options with the SQL Query Analyzer.

2. What is the best way to create and use Transact-SQL statements for future re-use?

   Use the Object Browser tool to script Transact-SQL statements directly from objects and from other templates. It is also possible to save a Transact-SQL script to a file for later modification and use.

3. How does a Transact-SQL batch differ from a Transact-SQL script?

   A Transact-SQL batch is a series of statements delineated by a GO statement that will be parsed and executed all at once.

   A Transact-SQL script is a file that contains one or more batches to be processed.