

Understanding T-SQL using Microsoft SQL Server

Instructor: Frank Stepanski

Overview:

Transact-SQL is central to the use of Microsoft® SQL Server. All applications that communicate with SQL Server do so by sending Transact-SQL statements to the server, regardless of an application's user interface. T-SQL (Transact-SQL) is a set of programming extensions from Microsoft that add several features to the Structured Query Language (SQL) including transaction control, exception and error handling, row processing, and declared variables.

[T-SQL](#) is a Microsoft proprietary language, although it is strongly linked with a standard set by the American National Standards Institute (ANSI). The current specification Microsoft bases its code on is ANSI-92.

Audience for this Class

This class is intended for analysts, developers, designers, administrators, and managers new to the SQL programming language.

Upon completion, students will understand SQL functions, join techniques, database objects and be able to write complex queries and stored procedures.

Tools Needed

The only tool needed for this class is [SQL Server 2008 Express](#). This is a free tool and has almost all of the features of the commercial version. The main differences are size limitations on databases (4 GB).

What is a Query?

A query is a way of retrieving some subset of information from a database. That information might be a single number such as a product price, a list of members with overdue subscriptions, or some sort of calculation such as the total amount of

products sold in the past 12 months. Once we retrieve this subset of data, we might want to update the database records or include the information in some sort of report.

Database Terminology

Before getting into the nuts and bolts of how to build queries, it is necessary to understand some of the ideas and terminology associated with [relational databases](#).

- *Tables*: These are where data is kept within the database. A database must contain at least one table to be of use, although you can have a database with no user tables and only system tables. **System tables** are special tables that SQL Server uses to help it work with the database. These tables contain information within rows and columns, much like in Excel, but they have a great deal more power than cells within Excel. **Temporary tables**—another type of database table—can take several different forms.
- *Columns*: These provide a definition of each single item of information that builds up to a table definition. A column is made up of cells that all hold data, much like a column in Excel. Unlike in Excel, though, where each cell can hold a different type of data, a column within a SQL Server table is restricted to what the data within it relates to, the type of data it will hold.
- *Rows*: A row is made up of one cell from every column defined for the table. There can be any number of rows in a table; you are limited only by your disk space, the amount of disk space that you defined as the maximum in your database creation definition, or the amount of disk space on your server. Rows are also called **records**.
- *Stored procedures*: When it comes to requiring a program to manipulate or work with data, or perform the same data-intensive task repeatedly, it's often better to store this code in a stored procedure. Stored procedures contain one or more T-SQL statements, which are compiled and ready to be executed when required. Stored procedures are permanently stored in the database, ready for use at any time.
- *T-SQL statements*: A T-SQL statement is a program statement that SQL Server can use to work with your data.

- *Indexes*: These can be regarded as predefined lists of information that can inform the database how the data is physically sorted and stored, or they can be used by SQL Server to find rows of data quickly using information supplied by a T-SQL query and matching this information to data within columns. An index consists of one or more columns from the table it is defined for.
- *Views*: These can be thought of as virtual tables. Views can contain information combined from several tables and can present a more user-friendly interface to the data. Views can also be indexed to speed processing of data within.

Downloading Microsoft SQL Server 2008 Express

Step 1: You need to [download and install](#) the *Runtime with Management Tools*. (Figure 1).

	Management Tools	Runtime	Runtime with Management Tools	Runtime with Advanced Services
SQL Server Management Database Engine		✓	✓	✓
SQL Server Management Studio Express	✓		✓	✓
Full-Text Search				✓
Reporting Services				✓
Download Size	172 MB	84 MB	224 MB	515 MB
Install	Install	Install	Install	Install

Figure 1 – Install Runtime with Management Tools option (3rd)

<http://www.microsoft.com/express/sql/download/>

Step 2: Before you can install the tools, you will be asked to install the [Microsoft Web Platform Installer](#).

Step 3: The installer program will appear (Figure 2).

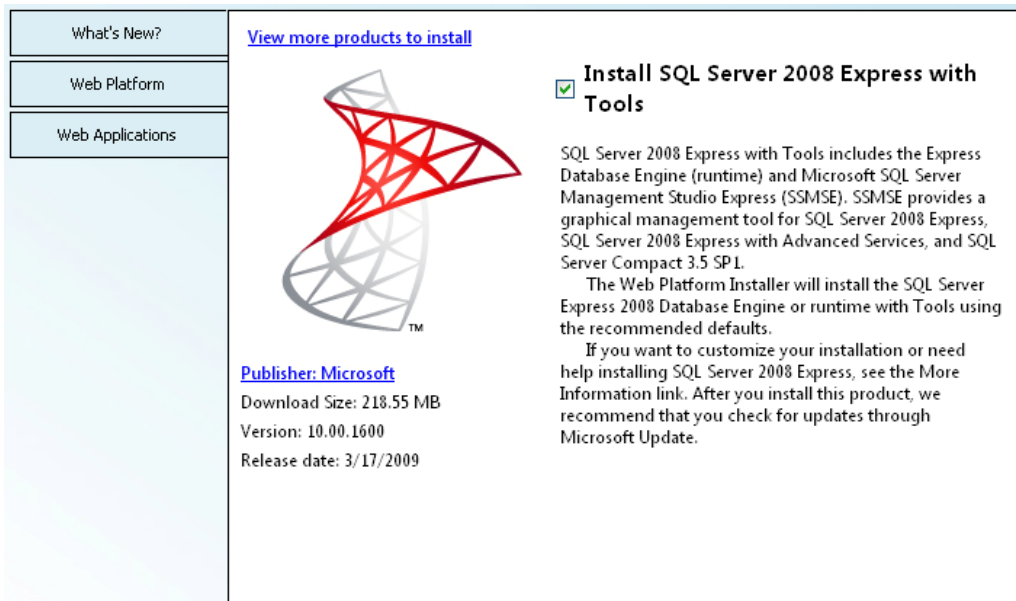


Figure 2 – Installer Screen

Step 4: You will be given two choices for [authentication mode](#) (Figure 3). For this class you will choose (default) the “Windows Integrated Authentication”, which will **not require** you to enter a username and password when you use SQL Server 2008 Express.

Windows Integrated Authentication mode requires users to provide a valid Windows username and password (if specified) to access the database server. In enterprise environments, these credentials are normally Active Directory domain credentials.

Mixed Authentication Mode allows the use of Windows credentials but supplements them with local SQL Server user accounts that the administrator may create and maintain within SQL Server.

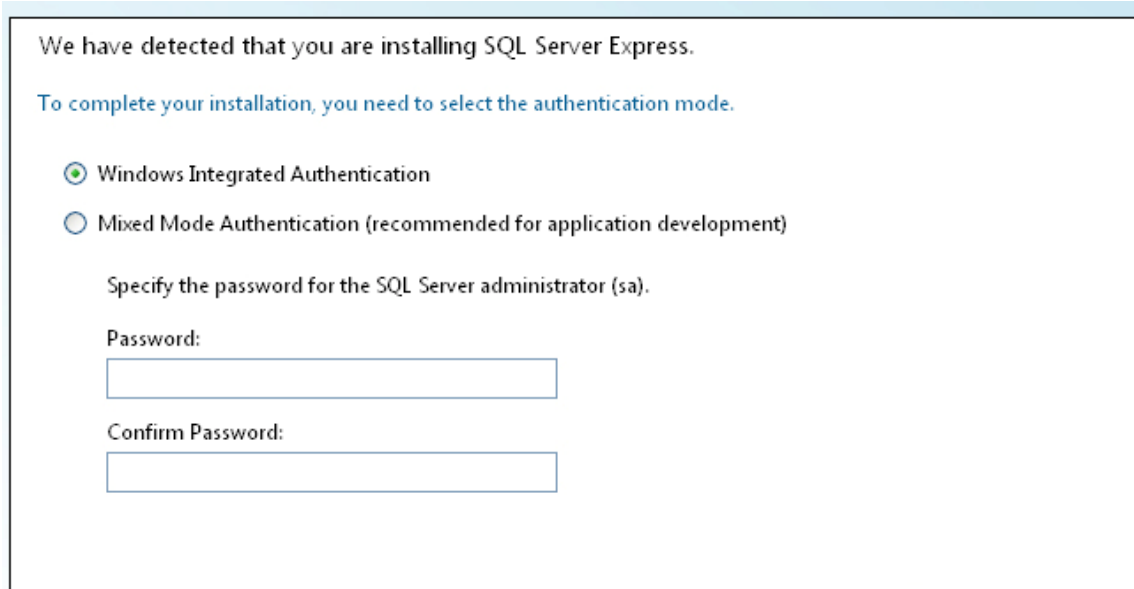


Figure 3 - Set Authentication Mode

Step 5: After installation has completed, you will get a few shortcuts added to your start menu. You will run **SQL Server Management Studio**.



SQL Server runs as a [Windows service](#). So what is a service? A good example of a service is any antivirus software that runs continuously from when the user restarts a computer to the point that the computer shuts down. A program, on the other hand, is either loaded in memory and running, or not started.

A service also has absolutely no user interface. There will be no form to display and no user input to deal with at run time. The only interaction with the process runs either through a separate user interface, which then links in to the service but is a totally separate unit of work.

This is why we also installed the visual GUI tool to SQL Server 2008 Express, [SQL Server Management Studio Express](#).

The sa Login

The sa login (sysadmin will be replacing it in future versions) is the default login that has full administration rights for SQL Server. If you had selected mixed mode authentication during the installation process, you would have seen that you would be forced to include a password for this account. This is because the sa user ID is such a powerful login.

It also exists in every SQL Server installation; therefore, any hacker knows that this user ID exists and so will try to connect to the server using it. Prior to SQL Server 2005 when creating a password became compulsory, many installations had the password blank, therefore allowing hackers instant access.

If you logged in to SQL Server as sa, you will have full control over any aspect of SQL Server. SQL Server inserts this ID no matter which authentication mode you install.



Figure 4 - Logging in to SQL Server 2008 Express

Now that SQL Server 2008 is successfully installed on your machine, it is time to explore the tool you will be using throughout this class.

SQL Server Management Studio Express

[SSMS](#) is the graphical user interface (GUI) you will use to run all of your queries and build your database solutions. SSMS helps you in the development of database solutions, including creating and modifying components of a database, amending the database itself, and dealing with security issues. This is an easy-to-use and intuitive tool, and before long, you will feel confident in using it to work with SQL Server quickly and efficiently.

SSMS (Figure 5) can even be used to develop and work with several installations of SQL Server in one application. These installations can be on one computer or on many computers connected through a local area network (LAN), a wide area network (WAN), or even the Internet.

Getting to know this tool well is crucial to becoming a successful professional SQL Server developer, as well as a database administrator.

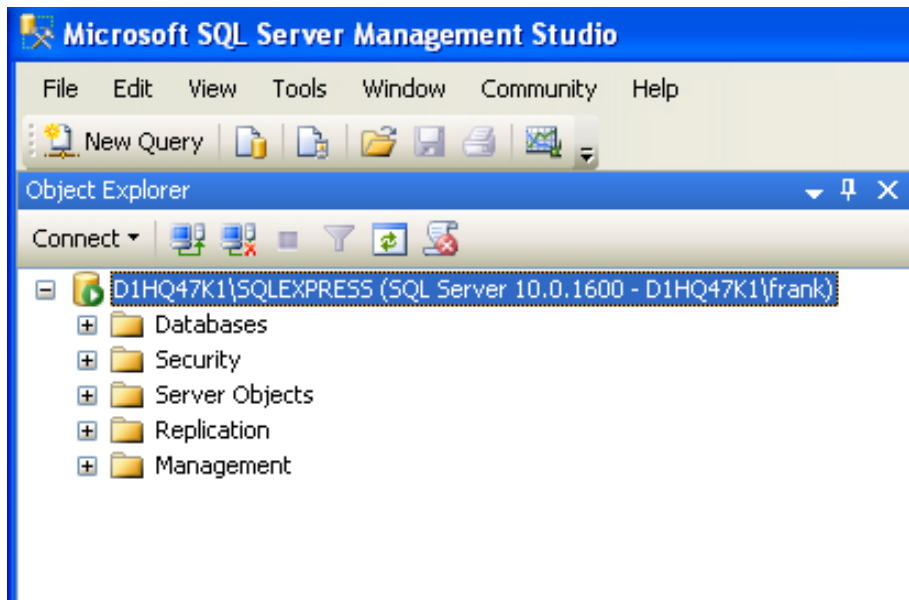


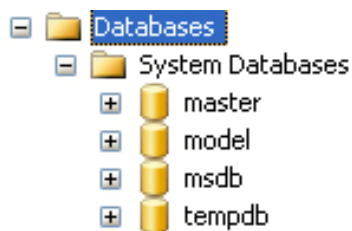
Figure 5 - SQL Server Management Studio Express

One of the tools within SSMS that we will use for completing tasks is Query Editor. This tool allows program code to be written and executed, from objects, to commands that manipulate data, and even complete tasks such as backing up data.

[Query Editor](#) is a tool within SSMS that allows you to programmatically build the same actions as dragging and dropping or using wizards. However, using T-SQL within Query Editor can give you more control over certain aspects of certain commands. Note that the name “Query Editor” comes from the fact that it sends **queries** to the database using T-SQL.

Prebuilt Databases Within SQL Server

Several databases are installed and displayed when SQL Server is first installed



Warning: Do not alter these databases!

master

The master database is at the heart of SQL Server, and if it should become corrupted, there is a very good chance that SQL Server will not work correctly.

The master database contains the following crucial information:

- All logins, or roles, that the user IDs belong to
- Every system configuration setting (e.g., data sorting information, security implementation, default language)
- The names of and information about the databases within the server
- The location of databases
- How SQL Server is initialized
- A list of the available languages
- System error and warning messages

tempdb

The tempdb database is—as its name suggests—a temporary database whose lifetime is the duration of a SQL Server session; once SQL Server stops, the

tempdb database is lost. When SQL Server starts up again, the tempdb database is re-created, fresh and new, and ready for use.

model

Whenever you create a database, it has to be modeled on a predefined set of criteria. For example, if you want all your databases to have a specific initial size or to have a specific set of information, you would place this information into the model database, which acts as a template database for further databases. If you want all databases to have a specific table within them, for example, then you would put this table in the model database.

The model database is used as the basis of the tempdb database.

msdb

msdb is another crucial database within SQL Server, as it provides the necessary information to run jobs to [SQL Server Agent](#). SQL Server Agent is a Windows service in SQL Server that runs any scheduled jobs that you set up (e.g., jobs that contain backup processing). A **job** is a process defined in SQL Server that runs automatically without any manual intervention to start it.

As with tempdb and model, you should not directly amend this database, and there is no real need to do so. Many other processes use msdb. For example, when you create a backup or perform a restore, msdb is used to store information about these tasks.

Installing the Sample Database

Let's install the database we will be using for the majority of the class.

In **wk1.zip**, you will find a file *TSQLFundamentals.sql* which includes all the statements to create the database we will use.

You can either just double click the file which will open the query editor in SSMS (Figure 6) or you can open the file by selecting *File* → *Open* → select the file.

```

TSQFundamentals.sql - D1...))
-----
-- Understanding T-SQL using Microsoft SQLServer
--
-- Script that creates the sample database TSQFundamentals
--
-- Supported versions of SQL Server: 2005, 2008
--
-- Based originally on the Northwind sample database
--
-----

-- Create Database

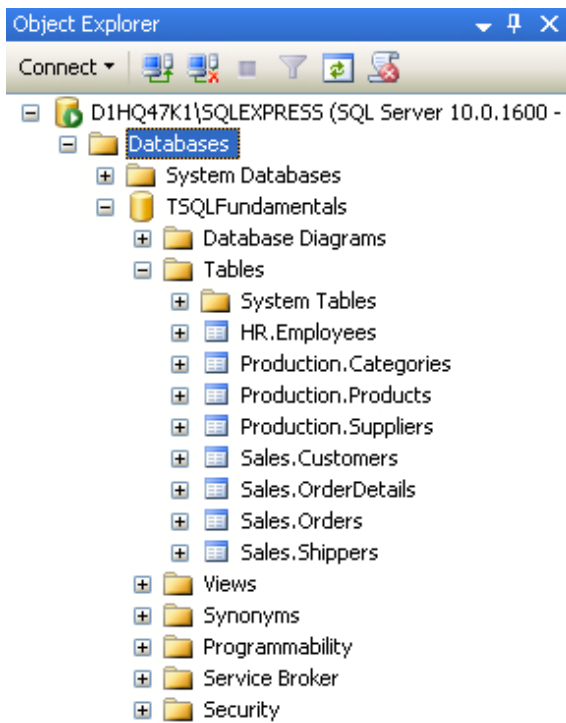
USE master;

-- Drop database
IF DB_ID('TSQFundamentals') IS NOT NULL DROP DATABASE TSQFundamentals;

-- If database could not be created due to open connections, abort

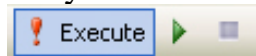
```

Figure 6 – Query Editor



Don't worry about trying to understand the statements in this file. You'll learn most of that in future lessons.

All you have to do now is execute this file:



You now have the TSQFundamentals database created. You can expand the Tables node to see all the tables that are in the database as well.

Note: You may have to refresh (F5) or re-Connect to see the new database.

What Is a Relational Database?

In simple terms, a relational database is a set of tables. Each table keeps information about aspects of one thing, such as a customer, an order, a product, a team, or a tournament. It is possible to set up constraints on the data in individual tables and also between tables.

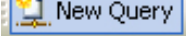
For example, when designing the database, we might specify that an order entered in the Order table must exist for a customer who exists in the Customer table.

Overview: Elements of the SELECT Statement

The purpose of a SELECT statement is to query tables, apply some logical manipulation, and return a result.

```
USE TSQLFundamentals;
```

```
SELECT empid, YEAR(orderdate) AS orderyear, COUNT(*) AS  
numorders  
FROM Sales.Orders  
WHERE custid = 71  
GROUP BY empid, YEAR(orderdate)  
HAVING COUNT(*) > 1  
ORDER BY empid, orderyear;
```

You will test this query by selecting New Query  which will bring up the Query Editor again. Then just execute it as before (Figure 7).

The code starts with a USE statement that sets the database context of your session to the TSQLFundamentals sample database. If your session is already in the context of the database you need to query, the USE statement is not required.

Before getting into the details of each phase of the SELECT statement, notice the order in which the query clauses are logically processed. In most programming languages the lines of code are processed in the order that they are written. In SQL things are different.

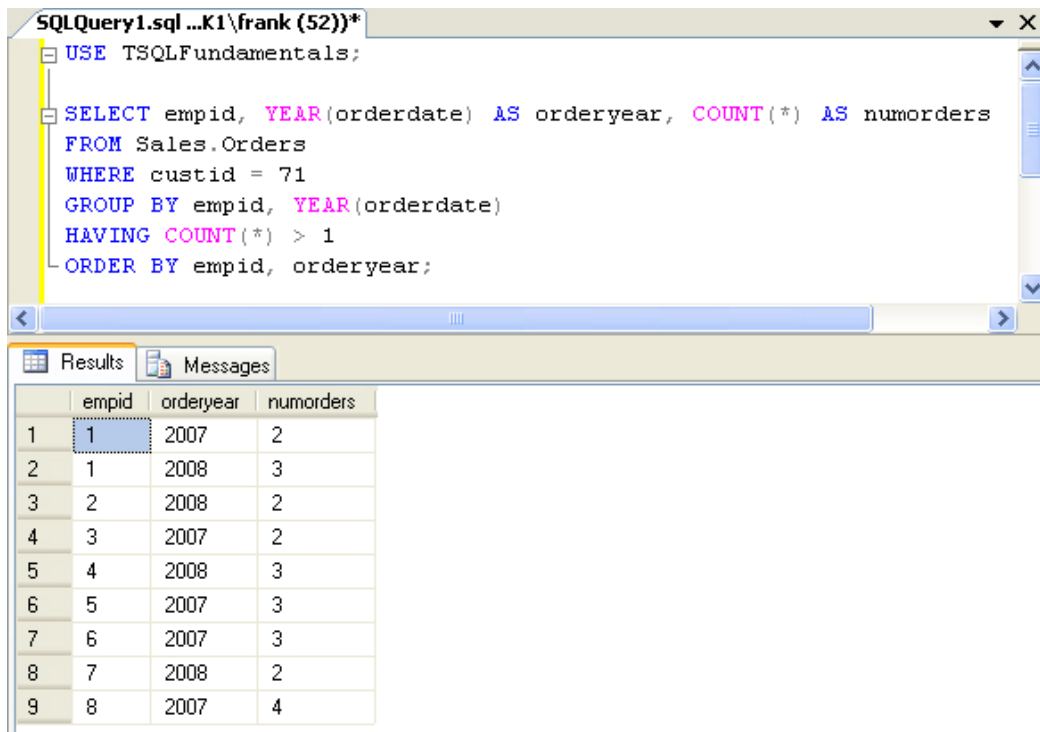


Figure 7 – Executing our first SELECT statement

Even though the SELECT clause appears first in the query, it is logically processed almost last. The clauses are logically processed in the following order:

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

So even though syntactically our sample query starts with a SELECT clause, logically its clauses are processed in the following order:

```

FROM Sales.Orders
WHERE custid = 71
GROUP BY empid, YEAR(orderdate)
HAVING COUNT(*) > 1
SELECT empid, YEAR(orderdate) AS orderyear, COUNT(*) AS
numorders ORDER BY empid, orderyear

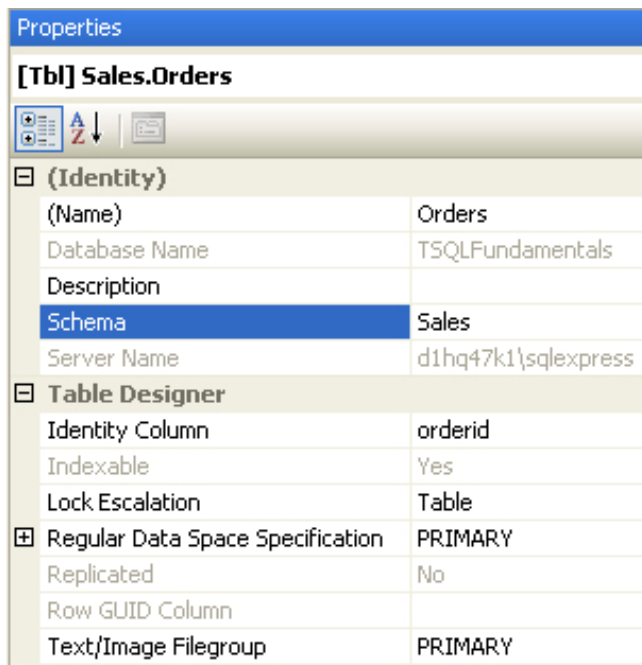
```

Or to present it in a more readable manner, here's what our statement does:

1. Queries the rows from the Sales.Orders table
2. Filters only orders where the customer ID is equal to 71
3. Groups the orders by employee ID and order year
4. Filters only groups (employee ID and order year) having more than one order
5. Selects (returns) for each group the employee ID, order year, and number of orders
6. Orders (sorts) the rows in the output by employee ID and order year

The FROM Clause

The FROM clause is the very first query clause that is logically processed. In this clause you specify the names of the tables that you want to query and table operators that operate on those tables. The FROM clause is simply where you specify the name of the table you want to query.



Properties	
[Tbl] Sales.Orders	
[Icons]	
(Identity)	
(Name)	Orders
Database Name	TSQLFundamentals
Description	
Schema	Sales
Server Name	d1hq47k1\sqlexpress
Table Designer	
Identity Column	orderid
Indexable	Yes
Lock Escalation	Table
Regular Data Space Specification	
Replicated	No
Row GUID Column	
Text/Image Filegroup	PRIMARY

When you don't specify the schema name (**Sales.Orders**) explicitly, SQL Server must resolve it implicitly. This creates some minor cost, and also leaves it to SQL Server to decide which object to use in case of ambiguity.

By being explicit, you ensure that you get the object that you intended to get, and that you don't pay any unnecessary penalties.

The WHERE Clause

In the WHERE clause, you specify a logical expression to filter the rows returned by the FROM phase. Only rows for which the logical expression evaluates to TRUE are returned by the WHERE phase to the subsequent logical query processing phase.

The GROUP BY Clause

The GROUP BY phase allows you to arrange the rows returned by the previous logical query processing phase in groups. The groups are determined by the elements you specify in the GROUP BY clause.

The HAVING Clause

With the HAVING clause you can specify a predicate/logical expression to filter groups as opposed to filtering individual rows, which happens in the WHERE phase. Only groups for which the logical expression in the HAVING clause evaluates to TRUE are returned by the HAVING phase to the next logical query processing phase. Groups for which the logical expression evaluates to FALSE or UNKNOWN are filtered out.

The ORDER BY Clause

The ORDER BY clause allows you to sort the rows in the output for presentation purposes. In terms of logical query processing, ORDER BY is the very last clause to be processed.

Note: A table has no guaranteed order, because a table is supposed to represent a set (or multiset if it has duplicates), and a set has no order. This means that when you query a table without specifying an ORDER BY clause, the query returns a table result, and SQL Server is free to return the rows in the output in any order. The only way for you to guarantee that the rows in the result are sorted is to explicitly specify an ORDER BY clause. When you want to sort by an expression in ascending order, you either specify ASC right after the expression, such as `orderyear ASC`. If you want to sort in descending order, you need to specify DESC after the expression, such as `orderyear DESC`.

Additional Resources:

Edition Features:

<http://www.microsoft.com/sqlserver/2008/en/us/editions-compare.aspx>

Download links:

<http://www.microsoft.com/sqlserver/2008/en/us/express.aspx>

Videos:

<http://www.msdev.com/Directory/SeriesDescription.aspx?CourseId=105>

SQL Server Magazine:

<http://www.sqlmag.com/>

SQL Server Search Engine:

<http://www.sqlhunt.com/>

Assignment for Lesson 1

1. Review the lesson.
2. [Download](#) and install SQL Server 2008 Express and SQL Server Management Studio Express.
3. Create the TSQLFundamentals database (wk1.zip).
4. Run the SELECT query (wk1.zip).
5. Try to modify the existing query to get some different results.
6. Post the results of your customized query on the class discussion board.

<http://webhost/lesson1/frank.html>

I will respond with a review on the class discussion board.

Copyright 2009 © Frank Stepanski

Used with Permission :: LVS Online Classes / LVS Associates

Lessons, files and content of these classes cannot be reproduced and/or published without the express written consent of the author.

Use of this site implies agreement with the [Terms of Use](#)