

## Developing SQL Queries for SQL Server®: Hands-On - 4 Days

*Course 534 Overview*

- You Will Learn How To**
- Develop complex and robust SQL queries for SQL Server 2008, 2005 and 2000
  - Query multiple tables with inner joins, outer joins and self joins
  - Transform data with built-in functions
  - Summarize data using aggregation and grouping
  - Execute analytic functions to calculate ranks
  - Build simple and correlated subqueries
- Course Benefits** Microsoft's Transact-SQL is the cornerstone of all SQL Server database operations. In this hands-on course, you learn to exploit the full potential of the SELECT statement to write robust queries using the best query method for your application, test your queries, and avoid common errors and pitfalls.
- Who Should Attend** Those who are developing systems using SQL Server databases, or who are using SQL to extract and analyze data from SQL Server databases. Knowledge of SQL Server at the level of Course 137, "SQL Server 2008 Comprehensive Introduction," or Course 925, "SQL Programming Language Introduction," is assumed.
- Hands-On Training** Learn by doing as you use Transact-SQL to solve problems and maximize performance. Instructor-led exercises include:
- Handling NULL values in expressions and conditions
  - Coding inner and outer joins
  - Implementing self joins
  - Computing aggregate results
  - Adding subtotals and grand totals to aggregate results
  - Employing ranking and analytic functions
  - Reusing subqueries as common table expressions
  - Analyzing query plans and tuning queries

## Developing SQL Queries for SQL Server®: Hands-On - 4 Days

Course 534 Outline

### Introduction and Overview

#### SQL fundamentals

- Retrieving data with SELECT
- Expressions
- Literals
- Handling NULLs properly

#### Executing queries

- Analyzing query plans
- Enhancing query performance
- Testing queries
- Selecting the best alternatives
- Avoiding errors and pitfalls

### Querying Multiple Tables

#### Implementing various types of joins

- Inner joins
- Cross joins
- Left, right and full outer joins
- Equijoins vs. theta joins
- The performance implications of joins
- Adding filter conditions to outer joins

#### Writing self joins

- Joining a table to itself
- Chaining self joins
- Solving time-interval problems

#### Combining queries with set operators

- UNION
- UNION ALL
- INTERSECT
- EXCEPT

### Scalar and Aggregate Functions

#### Taking advantage of scalar functions

- Converting datatypes
- Explicit vs. implicit conversion
- Performing calculations on dates and times
- Extracting date and time components
- Determining date and time format
- Manipulating strings
- Choosing the right function for the job

#### Summarizing data with aggregate functions

- COUNT
- SUM
- AVG
- MIN
- MAX
- Managing NULLs
- Suppressing duplicates

#### Grouping data

- GROUP BY and GROUP BY ALL
- Applying conditions with HAVING
- Calculating moving averages

#### Extending group queries

- Nesting grouped aggregates
- Joins and grouping

#### Building crosstab reports

- Using CASE to turn rows into columns
- Applying PIVOT

### Performing Extensive Analysis with Analytic Functions

#### The OVER clause

- Specifying the ordering before applying the function
- Splitting the result set into logical partitions

#### Calculating ranks

- RANK and DENSE\_RANK
- ROW\_NUMBER with ordered sets

#### Extending the use of aggregates

- Partitioning in multiple levels
- Comparing row and aggregate values

### Building Subqueries

#### Simple subqueries

- Subqueries in conditions and column expressions
- Creating multilevel subqueries
- Avoiding problems when subqueries return NULLs
- Handling multirow subquery results

#### Correlated subqueries

- Accessing values from the outer query
- EXISTS vs. IN
- Identifying duplicates
- Avoiding accidental correlation

#### Common table expressions

- Reusable and recursive subqueries
- Traversing hierarchies

### Breaking Down Complex Queries

- Overcoming SQL limitations
- Reducing complexity and improving performance

- Exploring alternatives for decomposing: temporary tables, views, common table expressions