

Microsoft Transact-SQL Programming

Duration: 5 Days (*Face-to-Face & Remote-Live*), or 35 Hours (*On-Demand*)

Price: \$2495 (*Face-to-Face & Remote-Live*), or \$1495 (*On-Demand*)

Discounts: We offer multiple discount options. [Click here](#) for more information.

Delivery Options: Attend face-to-face in the classroom, [remote-live](#) or [on-demand training](#).

Students Will Learn

- Designing normalized table structures for relational databases
- Creating databases and tables
- Using primary and foreign keys
- Writing SQL queries
- Using inner and outer joins
- Using set operators (UNION, INTERSECT, EXCEPT)
- Using DML for SELECT, INSERT, UPDATE, DELETE
- Using subqueries
- Using triggers and stored procedures
- Using aggregate functions to return totals and subtotals
- Programming features of T-SQL
- Table expressions
- T-SQL specific data types and functions
- T-SQL cursors

Course Description

This Transact-SQL programming course teaches students relational database fundamentals and SQL programming skills in the Microsoft SQL Server environment. Topics covered include relational database architecture, database design techniques, and simple and complex query skills. The course also covers Microsoft-specific T-SQL programming constructs, creation and use of stored procedures and user-defined functions, use of cursors and updateable views.

This class is intended for analysts, developers, designers, administrators, and managers new to the SQL programming language. Upon completion, participants will understand SQL functions, join techniques, database objects and constraints, and will be able to write useful stored procedures and views as well as complex queries and updates. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Course Prerequisites

Course Overview

Relational Database Fundamentals

- Overview of Relational Database Concepts
- Relational Databases and Relational Database Management Systems
- Data Normalization
- DDL Syntax

Creating a Database

- Database Development Methodology Overview
- Building a Logical Data Model
 - Identifying Entities and Attributes
 - Isolating Keys
 - Relationships Between Entities
 - Creating Entity-Relationship Diagrams
- Transforming to Physical Design
 - Migrating Entities to Tables
 - Selecting Primary Keys
 - Defining Columns
 - Enforcing Relationships with Foreign Keys
- Constructing the Database Using DDL
 - Creating Tables, Indexes, Constraints and Views
 - Dropping Tables, Indexes, Constraints and Views
 - Modifying Tables, Indexes, Constraints and Views

Advanced Query Techniques

- Inner Joins
- Outer Joins (Left, Right, Full)
- Performing Self-Joins
- Subqueries
 - Simple
 - Correlated
- Using the `EXISTS` Operator
- Tips for Developing Complex SQL Queries
- Using Aggregate Functions
 - `AVG`

Writing Basic SQL Queries

- Displaying Table Structures
- Retrieving Column Data From a Table or View
- Selecting Unique Values
- Filtering Rows Using the `WHERE` Clause
- Sorting Results Using `ORDER BY`
- Joining Multiple Tables
- Using Column and Table Aliases

Manipulating Query Results

- Using Row Functions
 - Character
 - Numeric
 - Date and Time
 - Data Conversion (`CAST` and `CONVERT`)
- Using the `CASE` Function
- Handling Null Values

Manipulating Table Data Using SQL's Data Manipulation Language (DML)

- Inserting Data into Tables
- Updating Existing Data
- Deleting Records
- Truncating Tables
- Performing Bulk Inserts
- Using the `OUTPUT` Clause
- Merging Data
- Working with Identity Columns and Sequences

- COUNT
 - SUM
 - MIN
 - MAX
- Performing Set Operations
 - UNION
 - INTERSECT
 - EXCEPT/MINUS
- Aggregating Results Using `GROUP BY`
- Restricting Groups with the `HAVING` Clause
- Creating Temporary Tables

User-Defined Functions

- Definition and Benefits of Use
- `CREATE FUNCTION`
 - Syntax
 - `RETURN` Clause and the `RETURNS` Statement
 - Scalar vs. Table Functions
- Comparison with Stored Procedures
- Returning Scalar Values and Tables
- `ALTER` and `DROP FUNCTION`

Triggers

- Definition and Benefits of Use
- Alternatives (e.g., Constraints)
- `CREATE TRIGGER`
 - Syntax
 - Trigger Types
- "Inserted" (or "NEW") and "Deleted" (or "OLD") Tables
- Event Handling and Trigger Execution
- `ALTER` and `DROP TRIGGER`

T-SQL Code Constructs

- Exploiting the Programming Features of T-SQL
 - Conditional Constructs
 - Looping Constructs
- Building Multi-Batch Scripts
- Invoking System Functions
- Using Variables in Scripts
- Using Temporary Tables in Scripts
- Handling Errors
 - Using `TRY...CATCH` Blocks
 - Using System Variables and Functions

Using T-SQL Cursors

- Overview of Cursors

Stored Procedures

- Definition and Benefits of Use
- `CREATE PROCEDURE`
 - Syntax
 - Defining Input Parameters
 - Defining Output Parameters
 - Defining Optional Parameters
- `ALTER` and `DROP PROCEDURE`
- Implementation Differences

Complex Queries

- Using Wildcard Characters with `Like`
- Allowing Users to Build SQL Queries Dynamically
- Pivoting Data
- Summarizing Data with `ROLLUP` and `CUBE`
- Using Partitioned Aggregates

Working with Data Types and Functions

- Effective Use of Data Types in SQL
 - String
 - Numeric
 - Time/Date
 - Other
- Substitution of Non-null Values with the `COALESCE` and `ISNULL` Functions
- Analyzing Data Points Using Ranking Functions

Working with Table Expressions

- Overview of Table Expressions

- Declaring a Cursor
- Using `OPEN CURSOR`, `CLOSE CURSOR`,
`DEALLOCATE CURSOR` Statements
- FETCHing Results
- Testing `@@FETCH_STATUS` and
`@@CURSOR_ROWS`
- Updating Records with Cursors
- Working with Views
- Using Derived Tables
- Common Table Expressions
- Table-Valued Functions

Hands On Technology Transfer
The Best Way to Transfer Technology Skills

1 Village Square, Suite 8
14 Fletcher Street
Chelmsford, MA 01824

Copyright © 2021 Hands On Technology Transfer, Inc.