Data Management Using Microsoft SQL Server

Session: 5

Transact-SQL

SQL Server 2012

Objectives

- Explain Transact-SQL
- List the different categories of Transact-SQL statements
- Explain the various data types supported by Transact-SQL
- Explain the Transact-SQL language elements
- Explain sets and predicate logic
- Describe the logical order of operators in the SELECT statement

Introduction

- SQL is the universal language used in the database world.
- Most modern RDBMS products use some type of SQL dialect as their primary query language.
- SQL can be used to create or destroy objects, such as tables, on the database server and to do things with those objects, such as put data into them or query for data.
- Transact-SQL is Microsoft's implementation of the standard SQL.
- Usually referred to as T-SQL, this language implements a standardized way to communicate to the database.
- The Transact-SQL language is an enhancement to SQL, the American National Standards Institute (ANSI) standard relational database language.
- It provides a comprehensive language that supports defining tables, inserting, deleting, updating, and accessing the data in the table.

Transact-SQL 1-2

Transact-SQL is a powerful language offering features such as data types, temporary objects, and extended stored procedures.

The Transact-SQL language in SQL Server 2012 provides improved performance, increased functionality, and enhanced features.

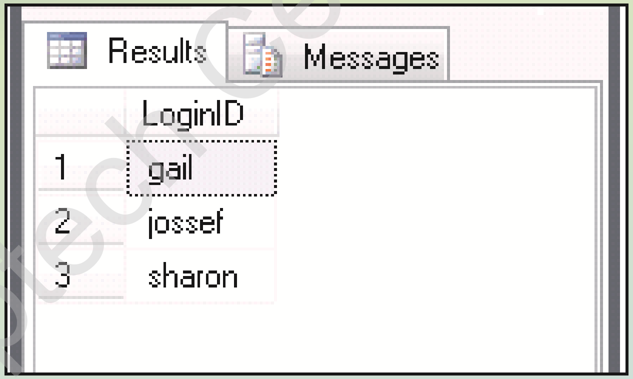
SQL Server 2012

Transact-SQL 2-2

Following code snippet shows the Transact-SQL statement, `SELECT`, which is used to retrieve all records of employees with 'Design Engineer' as the `JobTitle` from the `Employee` table.

```
SELECT LoginID
FROM Employee
WHERE JobTitle = 'Design Engineer'
```

- Following figure shows the result of the `SELECT` statement:



```
   LoginID
1  gail
2  jossef
3  sharon
```

- Transact-SQL includes many syntax elements that are used by or that influence most statements.
- These elements include data types, predicates, functions, variables, expressions, control-of-flow, comments, and batch separators.

SQL Server 2012

Data Definition Language (DDL)

DDL is used to define and manage all attributes and properties of a database, including row layouts, column definitions, key columns, file locations, and storage strategy.

For each object, there are usually CREATE, ALTER, and DROP statements (such as, CREATE TABLE, ALTER TABLE, and DROP TABLE).

CREATE object_name

ALTER object_name

DROP object_name

In DDL statements, object_name can be a table, view, trigger, stored procedure, and so on.

SQL Server 2012

Data Manipulation Language (DML)

DML is used to select, insert, update, or delete data in the objects defined with DDL.

The different DML statements are as follows:

SELECT statement

INSERT statement

UPDATE statement

DELETE statement

SQL Server 2012

Data Control Language (DCL)

Data is an important part of database, so proper steps should be taken to check that no invalid user accesses the data.

Permissions are controlled by using the GRANT, REVOKE, and DENY statements.

GRANT statement

REVOKE statement

DENY statement

SQL Server 2012

Data Types 1-6

- A data type is an attribute defining the type of data that an object can contain.
- Data types must be provided for columns, parameters, variables, and functions that return data values, and stored procedures that have a return code.
- Transact-SQL includes a number of base data types, such as `varchar`, `text`, and `int`.
- All data that is stored in SQL Server must be compatible with one of the base data types.
- The following objects have data types:

  - **Columns present in tables and views**
  - **Parameters in stored procedures**
  - **Variables**
  - **Transact-SQL functions that return one or more data values of a specific data type**
  - **Stored procedures that have a return code belonging to the integer data type**

- Various items in SQL Server 2012 such as columns, variables, and expressions are assigned data types.

Data Types 2-6

- SQL Server 2012 supports three kinds of data types:

System-defined Data Types

- These data types are provided by SQL Server 2012.
- Following table shows the commonly used system-defined data types of SQL Server 2012.

Category	Data Type	Description
Exact Numerics	int	A column of this type occupies 4 bytes of memory space. Is typically used to hold integer values. Can hold integer data from -2^31 (-2,147,483,648) to 2^31-1 (2,147,483,647).
	smallint	A column of this type occupies 2 bytes of memory space. Can hold integer data from –32,768 to 32,767.
	tinyint	A column of this type occupies 1 byte of memory space. Can hold integer data from 0 to 255.
	bigint	A column of this type occupies 8 bytes of memory space. Can hold data in the range -2^63 (-9,223,372,036,854,775,808) to 2^63-1 (9,223,372,036,854,775,807).
	numeric	A column of this type has fixed precision and scale.
	money	A column of this type occupies 8 bytes of memory space. Represents monetary data values ranging from –2^63/10000 (–922,337,203,685,477.5808) to 2^63–1 (922,337,203,685,477.5807).

SQL Server 2012

Data Types 3-6

Category	Data Type	Description
Approximate Numerics	float	A column of this type occupies 8 bytes of memory space. Represents floating point number ranging from –1.79E +308 through 1.79E+308.
	real	A column of this type occupies 4 bytes of memory space. Represents floating precision number ranging from –3.40E+38 through 3.40E+38.
Date and Time	datetime	Represents date and time. Stored as two 4-byte integers.
	smalldatetime	Represents date and time.
Character String	char	Stores character data that is fixed-length and non-Unicode.
	varchar	Stores character data that is variable-length and non-Unicode with a maximum of 8,000 characters.
	text	Stores character data that is variable-length and non-Unicode with a maximum length of 2^31 - 1 (2,147,483,647) characters.
Unicode Types	nchar	Stores Unicode character data of fixed-length.
	nvarchar	Stores variable-length Unicode character data.

SQL Server 2012

Data Types 4-6

Category	Data Type	Description
Other Data Types	timestamp	A column of this type occupies 8 bytes of memory space. Can hold automatically generated, unique binary numbers that are generated for a database.
	binary(n)	Stores fixed-length binary data with a maximum length of 8000 bytes.
	varbinary(n)	Stores variable-length binary data with a maximum length of 8000 bytes.
	image	Stores variable-length binary data with a maximum length of 2^30–1 (1,073,741,823) bytes.
	uniqueidentifier	A column of this type occupies 16 bytes of memory space. Also, stores a globally unique identifier (GUID).

Alias Data Types

- Are based on the system-supplied data types.
- Are used when more than one table stores the same type of data in a column and has similar characteristics such as length, nullability, and type.
- Can be created when it can be used commonly by all these tables.

SQL Server 2012

Data Types 5-6

- Alias data types can be created using the CREATE TYPE statement.
- The syntax for the CREATE TYPE statement is as follows:

Syntax:

```
CREATE TYPE [ schema_name. ] type_name { FROM base_type [ ( precision [ ,
 scale ] ) ] [ NULL | NOT NULL ] } [ ; ]
```

where,

schema_name: identifies the name of the schema in which the alias data type is being created. A schema is a collection of objects such as tables, views, and so forth in a database.

type_name: identifies the name of the alias type being created.

base_type: identifies the name of the system-defined data type based on which the alias data type is being created.

precision and scale: specify the precision and scale for numeric data.

NULL | NOT NULL: specifies whether the data type can hold a null value or not.

SQL Server 2012

Data Types 6-6

- Following code snippet shows how to create an alias data type using CREATE TYPE statement.

```
CREATE TYPE usertype FROM varchar(20) NOT NULL
```

- In the code, the built-in data type varchar is stored as a new data type named usertype by using the CREATE TYPE statement.

User-defined Types

- Are created using programming languages supported by the .NET Framework.

SQL Server 2012

Transact-SQL Language Elements

- Are used in SQL Server 2012 for working on data that is entered in SQL Server database.
- Includes the following elements:

- Predicates
- Operators
- Functions
- Variables
- Expressions
- Control-of Flow
- Errors
- Transactions
- Comments
- Batch Separators

Predicates and Operators 1-3

- Predicates are used to evaluate whether an expression is `TRUE`, `FALSE`, or `UNKNOWN`. Some of the predicates available in Transact-SQL are as follows:

IN	• Determines whether a specified value matches any value in a subquery or a list.
BETWEEN	• Specifies a range of values to test.
LIKE	• Used to match characters against a specified pattern.
CONTAINS	• Searches for precise or less precise matches to single words and phrases, words within a certain distance of one another, or weighted matches.

- Following table shows some examples of the predicates:

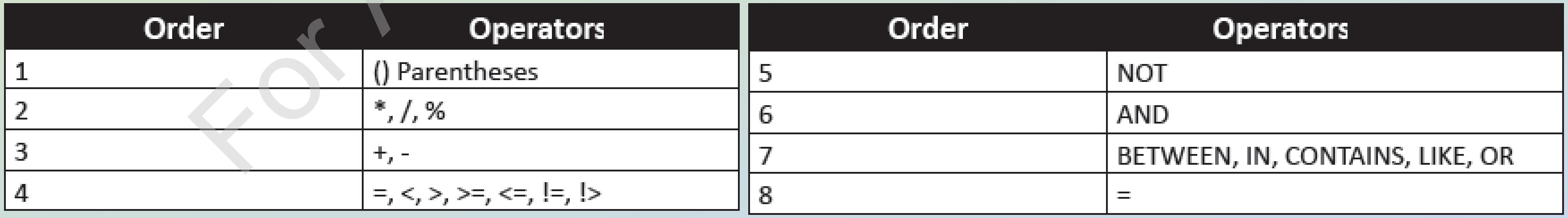
Predicate	Example
IN	SELECT UserID, FirstName, LastName, Salary FROM Employee WHERE Salary IN(5000,20000);
BETWEEN	Select UserID, FirstName, LastName, Salary FROM Employee WHERE Salary BETWEEN 5000 and 20000;
LIKE	Select UserID, FirstName, LastName, Salary FROM Employee WHERE FirstName LIKE '%h%‘
CONTAINS	SELECT UserID, FirstName, LastName, Salary FROM Employee WHERE Salary CONTAINS(5000);

Predicates and Operators 2-3

- Operators are used to perform arithmetic, comparison, concatenation, or assignment of values.
- SQL Server has seven categories of operators. Following table describes the different operators supported in SQL Server 2012.

Operator	Description	Example
Comparison	Compares a value against another value or an expression	=, <, >, >=, <=, !=, !>
Logical	Tests for the truth of a condition	AND, OR, NOT
Arithmetic	Performs arithmetic operations like addition, subtraction, multiplication, and division	+, -, *, /, %
Concatenation	Combines two strings into one string	+
Assignment	Assigns a value to a variable	=

- Following table shows the precedence of predicates and operators:



Order	Operators
1	() Parentheses
2	*, /, %
3	+, -
4	=, <, >, >=, <=, !=, !>
5	NOT
6	AND
7	BETWEEN, IN, CONTAINS, LIKE, OR
8	=

Predicates and Operators 3-3

- Following code snippet shows execution of operators according to precedence:

```
DECLARE @Number int;
SET @Number = 2 + 2 * (4 + (5 - 3))
SELECT @Number
```

- Here, the steps to arrive at the result are as follows:

1. 2 + 2 * (4 + (5 – 3))
2. 2 + 2 * (4 + 2)
3. 2 + 2 * 6
4. 2 + 12
5. 14

- Hence, the code will display 14.

SQL Server 2012

Functions 1-2

- Is a set of Transact-SQL statements that is used to perform some task.
- Four types of functions in SQL Server 2012 are as follows:

Rowset Functions

- Is used to return an object that can be used in place of a table reference.
- For example, `OPENDATASOURCE`, `OPENQUERY`, `OPENROWSET`, and `OPENXML`.

Aggregate Functions

- Provides aggregate functions to assist with the summarization of large volumes of data.
- For example, `SUM`, `MIN`, `MAX`, `AVG`, `COUNT`, `COUNTBIG`, and so on.

Ranking Functions

- Can implement many tasks, like creating arrays, generating sequential numbers, finding ranks, and so on in an easier and faster way by using ranking functions.
- For example, `RANK`, `DENSE_RANK`, `NTILE`, and `ROW_NUMBER`.

Scalar functions

- Input is a single value and the output received is also a single value.

SQL Server 2012

Functions 2-2

- Following table shows the commonly used scalar functions in SQL:

Function	Description	Example
Conversion function	The conversion function is used to transform a value of one data type to another. Additionally, it can be used to obtain a variety of special date formats.	CONVERT
Date and time function	Date and time functions are used to manipulate date and time values. They are useful to perform calculations based on time and dates.	GETDATE, SYSDATETIME, GETUTCDATE, DATEADD, DATEDIFF, YEAR, MONTH, DAY
Mathematical function	Mathematical functions perform algebraic operations on numeric values.	RAND, ROUND, POWER, ABS, CEILING, FLOOR
System function	SQL Server provides system functions for returning metadata or configuration settings.	HOST_ID, HOST_NAME, ISNULL
String function	String functions are used for string inputs such as char and varchar. The output can be a string or a numeric value.	SUBSTRING, LEFT, RIGHT, LEN, DATALENGTH, REPLACE, REPLICATE, UPPER, LOWER, RTRIM, LTRIM

- There are also other scalar functions such as cursor functions, logical functions, metadata functions, security functions, and so on that are available in SQL Server 2012.

SQL Server 2012

Variables

A variable is an object that can hold a data value. In Transact-SQL, variables can be classified into local and global variables.

Data can be passed to SQL statements using local variables. The name of a local variable must be prefixed with '`@`' sign.

The value of any of these variables can be retrieved with a simple `SELECT` query.

SQL Server 2012

Expressions 1-2

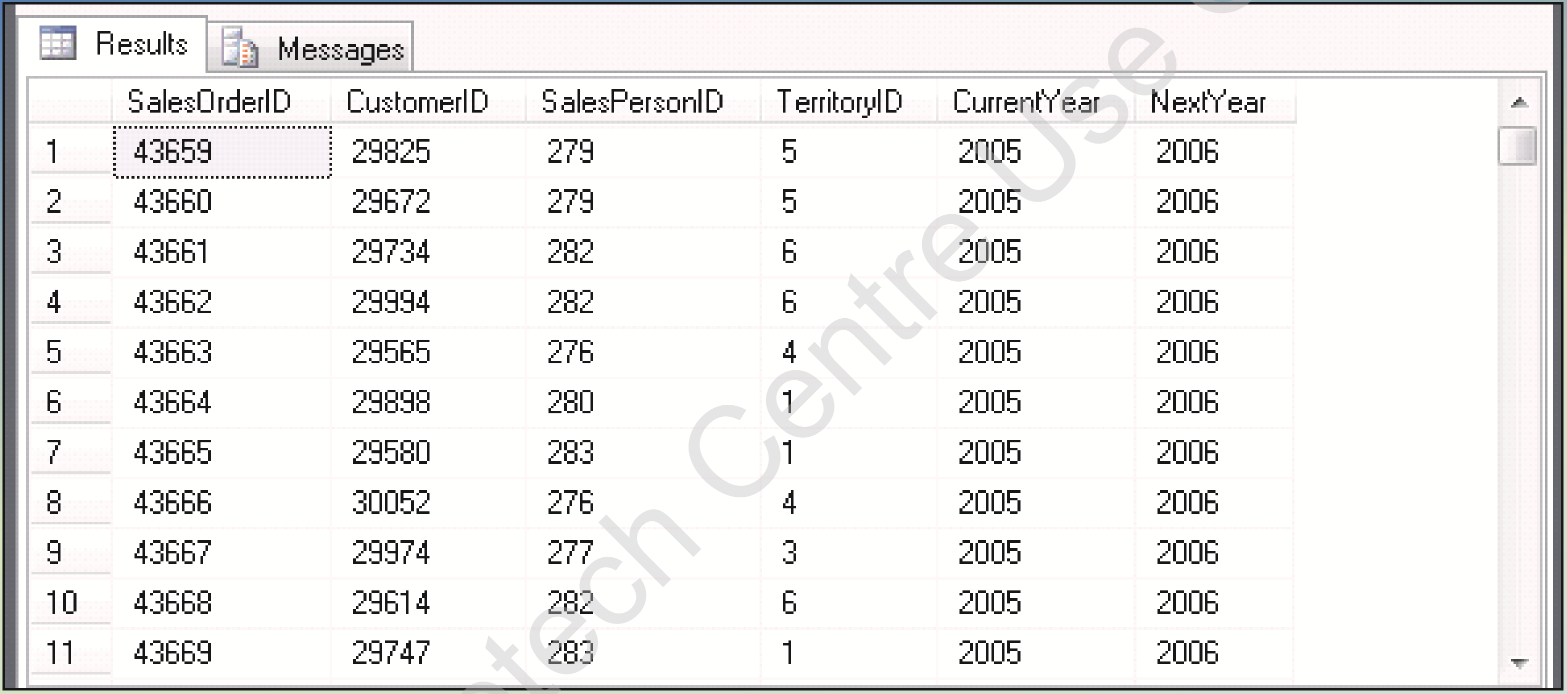
An expression is a combination of identifiers, values, and operators that SQL Server can evaluate in order to obtain a result.

Following code snippet shows an expression that operates on a column to add an integer to the results of the YEAR function on a datetime column:

```
SELECT SalesOrderID, CustomerID, SalesPersonID,
TerritoryID,YEAR(OrderDate)
AS CurrentYear, YEAR(OrderDate) + 1 AS NextYear
FROM Sales.SalesOrderHeader
```

Expressions 2-2

- Following figure shows the results of the expression:



Results | Messages

	SalesOrderID	CustomerID	SalesPersonID	TerritoryID	CurrentYear	NextYear
1	43659	29825	279	5	2005	2006
2	43660	29672	279	5	2005	2006
3	43661	29734	282	6	2005	2006
4	43662	29994	282	6	2005	2006
5	43663	29565	276	4	2005	2006
6	43664	29898	280	1	2005	2006
7	43665	29580	283	1	2005	2006
8	43666	30052	276	4	2005	2006
9	43667	29974	277	3	2005	2006
10	43668	29614	282	6	2005	2006
11	43669	29747	283	1	2005	2006

SQL Server 2012

Control-of-Flow, Errors, and Transactions

- Although Transact-SQL is primarily a data retrieval language, it supports control-of-flow statements for executing and finding errors.
- Control-of-flow language determines the execution flow of Transact-SQL statements, statement blocks, user-defined functions, and stored procedures.
- Following table shows some of the commonly used control-of-flow statements in Transact-SQL:

Control-of-Flow Statement	Description
IF. . .ELSE	Provides branching control based on a logical test.
WHILE	Repeats a statement or a block of statements as long as the condition is true.
BEGIN. . .END	Defines the scope of a block of Transact-SQL statements.
TRY. . . CATCH	Defines the structure for exception and error handling.
BEGIN TRANSACTION	Marks a block of statements as part of an explicit transaction.

Comments 1-2

- Comments are descriptive text strings, also known as remarks, in program code that will be ignored by the compiler.
- Comments can be included inside the source code of a single statement, a batch, or a stored procedure.
- Comments explain the purpose of the program, special execution conditions, and provide revision history information.
- Microsoft SQL Server supports two types of commenting styles:

- A complete line of code or a part of a code can be marked as a comment, if two hyphens (- -) are placed at the beginning.
- The remainder of the line becomes a comment.
- Following code snippet displays the use of this style of comment:

```
USE AdventureWorks2012
-- HumanResources.Employee table contains the details of an employee.
-- This statement retrieves all the rows of the table
-- HumanResources.Employee.
SELECT * FROM HumanResources.Employee
```

SQL Server 2012

Comments 2-2

- These comment characters can be used on the same line as code to be executed, on lines by themselves, or even within executable code.
- Everything in the line beginning from the open comment pair (/*) to the close comment pair (*/) is considered part of the comment.
- For a multiple-line comment, the open-comment character pair (/*) must begin the comment, and the close-comment character pair (*/) must end the comment.
- Following code snippet displays the use of this style of comment:

```
USE AdventureWorks2012
/* HumanResources.Employee table contains the details of an employee.
This statement retrieves all the rows of the table
HumanResources.Employee. */
SELECT * FROM HumanResources.Employee
```

SQL Server 2012

Batch Separators

A batch is a collection of one or more Transact-SQL statements sent at one time from an application to SQL Server for execution.

The statements in the execution plan are then executed one at a time.

A batch separator is handled by SQL Server client tools such as SSMS to execute commands. For example, you need to specify GO as a batch separator in SSMS.

- An example of a batch statement is given in the following code snippet:

```
USE AdventureWorks2012
SELECT * FROM HumanResources.Employee
GO
```

- Here, the two statements will be grouped into one execution plan but executed one statement at a time.
- The GO keyword signals the end of a batch.

Set Theory

Set theory is a mathematical foundation used in relational database model.

For example, all the employees under an `Employee` table can be considered as one set.

- Following table shows the different applications in the set theory and their corresponding application in SQL Server queries.

Set Theory Applications	Application in SQL Server Queries
Act on the whole set at once.	Query the whole table at once.
Use declarative, set-based processing.	Use attributes in SQL Server to retrieve specific data.
Elements in the set must be unique.	Define unique keys in the table.
No sorting instructions.	The results of querying are not retrieved in any order.

SQL Server 2012

Predicate Logic

- Predicate logic is a mathematical framework that consists of logical tests that gives a result. The results are always displayed as either true or false.
- In Transact-SQL, expressions such as `WHERE` and `CASE` expressions are based on predicate logic.
- Predicate logic is also used in other situations in Transact-SQL. Some of the applications of predicate logic in Transact-SQL are as follows:

- Enforcing data integrity using the **CHECK** constraint
- Control-of-flow using the **IF** statement
- Joining tables using the **ON** filter
- Filtering data in queries using the **WHERE** and **HAVING** clause
- Providing conditional logic to **CASE** expressions
- Defining subqueries

Logical Order of Operators in SELECT Statement 1-5

- Along with the syntax of different SQL Server elements, an SQL Server user must also know the process of how the entire query is executed.
- This process is a logical process that breaks the query and executes the query according to a predefined sequence in SQL Server 2012.
- The `SELECT` statement is a query that will be used to explain the logical process of query execution.
- The syntax of the `SELECT` statement is as follows:

Syntax:

```
SELECT <select list>
FROM <table source>
WHERE <search condition>
GROUP BY <group by list>
HAVING <search condition>
ORDER BY <order by list>
```

SQL Server 2012

Logical Order of Operators in SELECT Statement 2-5

- Following table explains the elements of the SELECT statement:

Element	Description
SELECT <select list>	Defines the columns to be returned
FROM <table source>	Defines the table to be queried
WHERE <search condition>	Filters the rows by using predicates
GROUP BY <group by list>	Arranges the rows by groups
HAVING <search condition>	Filters the groups using predicates
ORDER BY <order by list>	Sorts the output

- Following code snippet shows a SELECT statement:

```
SELECT SalesPersonID, YEAR(OrderDate) AS OrderYear
FROM Sales.SalesOrderHeader
WHERE CustomerID = 30084
GROUP BY SalesPersonID, YEAR(OrderDate)
HAVING COUNT(*) > 1
ORDER BY SalesPersonID, OrderYear;
```

SQL Server 2012

Logical Order of Operators in SELECT Statement 3-5

- In the example, the order in which SQL Server will execute the `SELECT` statement is as follows:

1
- First, the `FROM` clause is evaluated to define the source table that will be queried.

2
- Next, the `WHERE` clause is evaluated to filter the rows in the source table.
- This filtering is defined by the predicate mentioned in the `WHERE` clause.

3
- After this, the `GROUP BY` clause is evaluated.
- This clause arranges the filtered values received from the `WHERE` clause.

4
- Next, the `HAVING` clause is evaluated based on the predicate that is provided.

5
- Next, the `SELECT` clause is executed to determine the columns that will appear in the query results.

6
- Finally, the `ORDER BY` statement is evaluated to display the output.

Logical Order of Operators in SELECT Statement 4-5

- The order of execution for the `SELECT` statement in the code snippet would be as follows:

5
- `SELECT SalesPersonID, YEAR(OrderDate) AS OrderYear`

1
- `FROM SalesOrderHeader`

2
- `WHERE CustomerID = 30084`

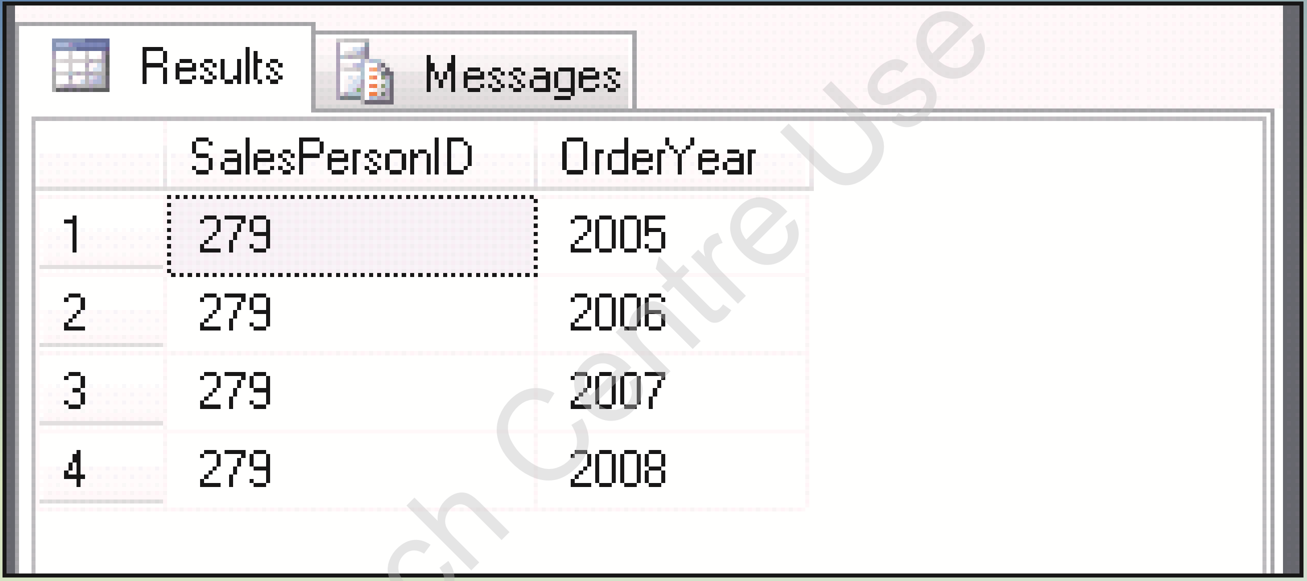
3
- `GROUP BY SalesPersonID, YEAR(OrderDate)`

4
- `HAVING COUNT(*) > 1`

6
- `ORDER BY SalesPersonID, OrderYear;`

Logical Order of Operators in SELECT Statement 5-5

- Following figure shows the result of the `SELECT` statement:



Results | Messages

	SalesPersonID	OrderYear
1	279	2005
2	279	2006
3	279	2007
4	279	2008

Summary

- Transact-SQL is a powerful language which offers features like data types, temporary objects, and extended stored procedures.
- SQL Server supports three types of Transact-SQL statements, namely, DDL, DML, and DCL.
- A data type is an attribute defining the type of data that an object can contain.
- The Transact-SQL language elements includes predicates, operators, functions, variables, expressions, control-of-flow, errors, and transactions, comments, and batch separators.
- Sets and Predicate Logic are the two mathematical fundamentals that are used in SQL Server 2012.
- Set theory is a mathematical foundation used in relational database model, where a set is a collection of distinct objects considered as a whole.
- Predicate logic is a mathematical framework that consists of logical tests that gives a result.