# Module 5

Sorting and Filtering Data

## Module Overview

- Sorting Data
- Filtering Data with Predicates
- Filtering Data with TOP and OFFSET-FETCH
- Working with Unknown Values

## Lesson 1: Sorting Data

- Using the ORDER BY Clause
- ORDER BY Clause Syntax
- ORDER BY Clause Examples
- Demonstration: Sorting Data

# Using the ORDER BY Clause

- ORDER BY sorts rows in results for presentation purposes
  - No guaranteed order of rows without ORDER BY
  - Use of ORDER BY guarantees the sort order of the result
  - Last clause to be logically processed
  - Sorts all NULLs together
- ORDER BY can refer to:
  - Columns by name, alias or ordinal position (not recommended)
  - Columns not part of SELECT list
    - Unless DISTINCT specified
- Declare sort order with ASC or DESC

## ORDER BY Clause Syntax

Writing ORDER BY using column names:

```
SELECT <select list>
FROM <table source>
ORDER BY <column1_name>, <column2_name>;
```

Writing ORDER BY using column aliases:

```
SELECT <column> AS <alias>
FROM <table source>
ORDER BY <alias1>, <alias2>;
```

- Specifying sort order in the ORDER BY clause:

```
SELECT <column> AS <alias>
FROM <table source>
ORDER BY <column_name|alias> ASC|DESC;
```

## ORDER BY Clause Examples

- ORDER BY with column names:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate;
```

- ORDER BY with column alias:

```
SELECT orderid, custid, YEAR(orderdate) AS orderyear
FROM Sales.Orders
ORDER BY orderyear;
```

- ORDER BY with descending order:

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
ORDER BY orderdate DESC;
```

## Demonstration: Sorting Data

In this demonstration, you will see how to:

- Sort data using the ORDER BY clause

## Lesson 2: Filtering Data with Predicates

- Filtering Data in the WHERE Clause with Predicates
- WHERE Clause Syntax
- Demonstration: Filtering Data with Predicates

# Filtering Data in the WHERE Clause with Predicates

- WHERE clauses use predicates
  - Must be expressed as logical conditions
  - Only rows for which predicate evaluates to TRUE are accepted
  - Values of FALSE or UNKNOWN filtered out
- WHERE clause follows FROM, precedes other clauses
  - Can't see aliases declared in SELECT clause
- Can be optimized by SQL Server to use indexes
- Data filtered server-side
  - Can reduce network traffic and client memory usage

# WHERE Clause Syntax

- Filter rows for customers from Spain

```
SELECT contactname, country
FROM Sales.Customers
WHERE country = N'Spain';
```

- Filter rows for orders after July 1, 2007

```
SELECT orderid, orderdate
FROM Sales.Orders
WHERE orderdate > '20070101';
```

- Filter orders within a range of dates

```
SELECT orderid, custid, orderdate
FROM Sales.Orders
WHERE orderdate >= '20070101' AND orderdate < '20080101';
```

## Demonstration: Filtering Data with Predicates

In this demonstration, you will see how to:

- Filter data in a WHERE clause

## Lesson 3: Filtering Data with TOP and OFFSET-FETCH

- Filtering in the SELECT Clause Using the TOP Option
- Filtering in the ORDER BY Clause Using OFFSET-FETCH
- OFFSET-FETCH Syntax
- Demonstration: Filtering Data with TOP and OFFSET-FETCH

# Filtering in the SELECT Clause Using the TOP Option

- TOP allows you to limit the number or percentage of rows returned by a query
- Works with ORDER BY clause to limit rows by sort order:
  - If ORDER BY list is not unique, results are not deterministic (no single correct result set)
  - Modify ORDER BY list to ensure uniqueness, or use TOP WITH TIES
- Added to SELECT clause:
  - SELECT TOP (N) | TOP (N) Percent
    - With percent, number of rows rounded up (nondeterministic)
  - SELECT TOP (N) WITH TIES
    - Retrieve duplicates where applicable (deterministic)
- TOP is proprietary to Microsoft SQL Server

## Filtering in the ORDER BY Clause Using OFFSET-FETCH

OFFSET-FETCH is an extension to the ORDER BY clause:

- Allows filtering a requested range of rows
  - Dependent on ORDER BY clause
- Provides a mechanism for paging through results
- Specify number of rows to skip, number of rows to retrieve:

```
ORDER BY <order_by_list>
OFFSET <offset_value> ROW(S)
FETCH FIRST|NEXT <fetch_value> ROW(S) ONLY
```

- Available in SQL Server 2012, 2014, and 2016
  - Provides more compatibility than TOP

## OFFSET-FETCH Syntax

- OFFSET value must be supplied
  - May be zero if no skipping is required
- The optional FETCH clause allows all rows following the OFFSET value to be returned
- Natural Language approach to code:
  - ROW and ROWS interchangeable
  - FIRST and NEXT interchangeable
  - ONLY optional—makes meaning clearer to human reader
- OFFSET value and FETCH value may be constants or expressions, including variables and parameters

```
OFFSET <offset_value> ROW|ROWS
FETCH FIRST|NEXT <fetch_value> ROW|ROWS [ONLY]
```

## Demonstration: Filtering Data with TOP and OFFSET-FETCH

In this demonstration, you will see how to:

- Filter data using TOP and OFFSET-FETCH

## Lesson 4: Working with Unknown Values

- Three-Valued Logic
- Handling NULL in Queries
- Demonstration: Working with NULL

## Three-Valued Logic

- SQL Server uses NULLs to mark missing values
  - NULL can be "missing but applicable" or "missing but inapplicable"
    - Customer middle name: Not supplied, or doesn’t have one?
- With no missing values, predicate outputs are TRUE or FALSE only ( 5 > 2, 1=1)
- With missing values, outputs can be TRUE, FALSE or UNKNOWN (NULL > 99, NULL = NULL)
- Predicates return UNKNOWN when comparing missing value to another value, including another missing value

## Handling NULL in Queries

- Different components of SQL Server handle NULL differently
  - Query filters (ON, WHERE, HAVING) filter out UNKNOWNs
  - CHECK constraints accept UNKNOWNS
  - ORDER BY, DISTINCT treat NULLs as equals
- Testing for NULL
  - Use IS NULL or IS NOT NULL rather than = NULL or <> NULL

```
SELECT custid, city, region, country
FROM Sales.Customers
WHERE region IS NOT NULL;
```

## Demonstration: Working with NULL

In this demonstration, you will see how to:

- Test for NULL

## Lab: Sorting and Filtering Data

- Exercise 1: Write Queries that Filter Data Using a WHERE Clause
- Exercise 2: Write Queries that Sort Data Using an ORDER BY Clause
- Exercise 3: Write Queries that Filter Data Using the TOP Option
- Exercise 4: Write Queries that Filter Data Using the OFFSET-FETCH Clause

**Logon Information**
Virtual machine: **20761C-MIA-SQL**
User name: **ADVENTUREWORKS\Student**
Password: **Pa55w.rd**

**Estimated Time: 60 minutes**

## Lab Scenario

You are an Adventure Works business analyst who will be writing reports using corporate databases stored in SQL Server. You have been provided with a set of data business requirements and will write T-SQL queries to retrieve the specified data from the databases. You will need to retrieve only some of the available data, and return it to your reports in a specified order.

## Lab Review

- What is the difference between filtering using the TOP option, and filtering using the WHERE clause?

## Module Review and Takeaways

- Review Question(s)