Session 3

10/28/2024

Module 3

Writing SELECT Queries

رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Module Overview

- Writing Simple SELECT Statements
- Eliminating Duplicates with DISTINCT
- Using Column and Table Aliases
- Writing Simple CASE Expressions

روّاد مصر الرقمية

وزارة الاتصـــــــــالات وتكنولوجيا المعلومات

Lesson 1: Writing Simple SELECT Statements

- Elements of the SELECT Statement
- Retrieving Columns from a Table or View
- Displaying Columns
- Using Calculations in the SELECT Clause
- Demonstration: Writing Simple SELECT Statements

Elements of the SELECT Statement

Clause	Expression
SELECT	<select list>
FROM	<table or view>
WHERE	<search condition>
GROUP BY	<group by list>
HAVING	<search condition>
ORDER BY	<order by list>

Retrieving Columns from a Table or View

- Use SELECT with column list to show columns
- Use FROM to specify the source table or view
  - Specify both schema and object names
- Delimit names if necessary
- End all statements with a semicolon
- In this example we use 'Sales' DB with 'Customers' table.

Keyword	Expression
SELECT	<select list>
FROM	<table or view>

```
SELECT companyname, country
FROM Sales.Customers;
```

Displaying Columns

- Displaying all columns
  - This is not best practice in production code!

```
SELECT *
FROM Sales.Customers;
```

- Displaying only specified columns

```
SELECT companyname, country
FROM Sales.Customers;
```

Using Calculations in the SELECT Clause

- Calculations are scalar, returning one value per row

Operator	Description
+	Add or concatenate
-	Subtract
*	Multiply
/	Divide
%	Modulo

- Using scalar expressions in the SELECT clause

```
SELECT unitprice, qty, (qty * unitprice)
FROM Sales.OrderDetails;
```

رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Demonstration: Writing Simple SELECT Statements

In this demonstration you will see how to:

- Use simple SELECT queries

Lesson 2: Eliminating Duplicates with DISTINCT

- SQL Sets and Duplicate Rows
- Understanding DISTINCT
- SELECT DISTINCT Syntax
- Demonstration: Eliminating Duplicates with DISTINCT

SQL Sets and Duplicate Rows

- SQL query results are not truly relational:
  - Rows are not guaranteed to be unique
  - No guaranteed order
- Even unique rows in a source table can return duplicate values for some columns

```
SELECT country
FROM Sales.Customers;
```

Query result:

```
country
-----------
Argentina
Argentina
Belgium
Austria
Austria
```

Understanding DISTINCT

- DISTINCT specifies that only unique rows can appear in the result set
- Removes duplicates based on column list results, not source table
- Provides uniqueness across set of selected columns
- Removes rows already operated on by WHERE, HAVING, and GROUP BY clauses
- Some queries may improve performance by filtering out duplicates before execution of SELECT clause

SELECT DISTINCT Syntax

```
SELECT DISTINCT <column list>

FROM <table or view>
```

```
SELECT DISTINCT companyname, country
FROM Sales.Customers;
```

```
companyname         country
--------------      -------

Customer AHPOP      UK
Customer AHXHT      Mexico
Customer AZJED      Germany
Customer BSVAR      France
Customer CCFIZ      Poland
```

Demonstration: Eliminating Duplicates with DISTINCT

In this demonstration, you will see how to:

- Eliminate duplicate rows

Lesson 3: Using Column and Table Aliases

- Use Aliases to Refer to Columns
- Use Aliases to Refer to Tables
- The Impact of Logical Processing Order on Aliases
- Demonstration: Using Column and Table Aliases

Use Aliases to Refer to Columns

- Column aliases using AS

```
SELECT orderid, unitprice, qty AS quantity
FROM Sales.OrderDetails;
```

- Column aliases using =

```
SELECT orderid, unitprice, quantity = qty
FROM Sales.OrderDetails;
```

- Accidental column aliases

```
SELECT orderid, unitprice quantity
FROM Sales.OrderDetails;
```

رواد مصر الرقمية

وزارة الاتصـــــــالات
وتكنولوجيا المعلومات

Use Aliases to Refer to Tables

- Create table aliases in the FROM clause
- Create table aliases with AS

```
SELECT custid, orderdate
FROM SalesOrders AS SO;
```

- Create table aliases without AS

```
SELECT custid, orderdate
FROM SalesOrders SO;
```

- Using table aliases in the SELECT clause

```
SELECT SO.custid, SO.orderdate
FROM SalesOrders AS SO
```

The Impact of Logical Processing Order on Aliases

- FROM, WHERE, and HAVING clauses processed before SELECT
- Aliases created in SELECT clause only visible to ORDER BY
- Expressions aliased in SELECT clause may be repeated elsewhere in query

```sql
SELECT EmployeeId, OrderDate AS OrderYear
FROM Sales.Orders
WHERE CustomerId = 71
HAVING COUNT(*) > 1
ORDER BY EmployeeId, OrderYear;
```

Demonstration: Using Column and Table Aliases

In this demonstration, you will see how to:

- Use column and table aliases

Lesson 4: Writing Simple CASE Expressions

- Using CASE Expressions in SELECT Clauses
- Forms of CASE Expressions
- Demonstration: Simple CASE Expressions

Using CASE Expressions in SELECT Clauses

- T-SQL CASE expressions return a single (scalar) value
- CASE expressions may be used in:
  - SELECT column list
  - WHERE or HAVING clauses
  - ORDER BY clause
- CASE returns result of expression
  - Not a control-of-flow mechanism
- In SELECT clause, CASE behaves as calculated column requiring an alias

Forms of CASE Expressions

- Two forms of T-SQL CASE expressions:
- Simple CASE
  - Compares one value to a list of possible values
  - Returns first match
  - If no match, returns value found in optional ELSE clause
  - If no match and no ELSE, returns NULL
- Searched CASE
  - Evaluates a set of predicates, or logical expressions
  - Returns value found in THEN clause matching first expression that evaluates to TRUE

Examples

```
SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
    WHEN City IS NULL THEN Country
    ELSE City
END);
```

```
SELECT OrderID, Quantity,
CASE
    WHEN Quantity > 30 THEN 'The quantity is greater than 30'
    WHEN Quantity = 30 THEN 'The quantity is 30'
    ELSE 'The quantity is under 30'
END AS QuantityText
FROM OrderDetails;
```

رواد مصر الرقمية

وزارة الاتصـــــالات
وتكنولوجيا المعلومات

Demonstration: Simple CASE Expressions

In this demonstration, you will see how to:

- Use a simple CASE expression

رواد مصر الرقمية

وزارة الاتصـــــــالات وتكنولوجيا المعلومات

Best Practices for Writing SQL Queries

- Use Uppercase for Keywords
- Use Lowercase or Snake Case for Names
- Use Table Aliases when Querying Multiple Tables
- Use Descriptive and Concise Aliases
- Avoid Writing SELECT *
- Use INNER JOIN Query instead of the WHERE Clause for Joining Tables
- Use the DISTINCT clause to get unique Results
- Use the LIMIT clause to Reduce Search Results

رواد مصر الرقمية

وزارة الاتصـــــالات وتكنولوجيا المعلومات

Lab: Writing Basic SELECT Statements

- Exercise 1: Writing Simple SELECT Statements
- Exercise 2: Eliminating Duplicates Using DISTINCT
- Exercise 3: Using Table and Column Aliases
- Exercise 4: Using a Simple CASE Expression

Lab Scenario

As a business analyst for Adventure Works, you will be writing reports using corporate databases stored in SQL Server. You can use your set of business requirements for data to write basic T-SQL queries to retrieve the specified data from the databases.

Module Review and Takeaways

- Review Question(s)
- Real-world Issues and Scenarios
- Best Practice