**SQL Introduction – 6**

**DML (Data Manipulation Language)**

## The IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

### SQL IN Syntax

SELECT column\_name(s)
FROM table\_name
WHERE column\_name IN (value1,value2,...);

## Example

SELECT \* FROM Customers
WHERE City IN ('Paris','London');

## The SQL BETWEEN Operator

The BETWEEN operator selects values within a range. The values can be numbers, text, or dates.

### SQL BETWEEN Syntax

SELECT column\_name(s)
FROM table\_name
WHERE column\_nameBETWEEN value1 AND value2;

Below is a selection from the "Products" table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ProductID** | **ProductName** | **SupplierID** | **CategoryID** | **Unit** | **Price** |
| 1 | Chais | 1 | 1 | 10 boxes x 20 bags | 18 |
| 2 | Chang | 1 | 1 | 24 - 12 oz bottles | 19 |
| 3 | Aniseed Syrup | 1 | 2 | 12 - 550 ml bottles | 10 |
| 4 | Chef Anton's Cajun Seasoning | 1 | 2 | 48 - 6 oz jars | 22 |
| 5 | Chef Anton's Gumbo Mix | 1 | 2 | 36 boxes | 21.35 |

The following SQL statement selects all products with a price BETWEEN 10 and 20:

## Example

SELECT \* FROM Products
WHERE Price BETWEEN 10 AND 20;

## NOT BETWEEN Operator Example

To display the products outside the range of the previous example, use NOT BETWEEN:

## Example

SELECT \* FROM Products
WHERE Price NOT BETWEEN 10 AND 20;

## BETWEEN Operator with IN Example

The following SQL statement selects all products with a price BETWEEN 10 and 20, but products with a CategoryID of 1,2, or 3 should not be displayed:

## Example

SELECT \* FROM Products
WHERE (Price BETWEEN 10 AND 20)
AND NOT CategoryID IN (1,2,3);

## BETWEEN Operator with Text Value Example

The following SQL statement selects all products with a ProductName beginning with any of the letter BETWEEN 'C' and 'M':

## Example

SELECT \* FROM Products
WHERE ProductName BETWEEN 'C' AND 'M';

## NOT BETWEEN Operator with Text Value Example

The following SQL statement selects all products with a ProductName beginning with any of the letter NOT BETWEEN 'C' and 'M':

## Example

SELECT \* FROM Products
WHERE ProductName NOT BETWEEN 'C' AND 'M';

## Sample Table

Below is a selection from the "Orders" table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OrderID** | **CustomerID** | **EmployeeID** | **OrderDate** | **ShipperID** |
| 10248 | 90 | 5 | 7/4/1996 | 3 |
| 10249 | 81 | 6 | 7/5/1996 | 1 |
| 10250 | 34 | 4 | 7/8/1996 | 2 |
| 10251 | 84 | 3 | 7/9/1996 | 1 |
| 10252 | 76 | 4 | 7/10/1996 | 2 |

## BETWEEN Operator with Date Value Example

The following SQL statement selects all orders with an OrderDate BETWEEN '04-July-1996' and '09-July-1996':

## Example

SELECT \* FROM Orders
WHERE OrderDate BETWEEN #07/04/1996# AND #07/09/1996#;

## SQL Aliases

SQL aliases are used to give a database table, or a column in a table, a temporary name.

Basically aliases are created to make column names more readable.

### SQL Alias Syntax for Columns

SELECT column\_name AS alias\_name
FROM table\_name;

### SQL Alias Syntax for Tables

SELECT column\_name(s)
FROM table\_nameAS alias\_name;

## Alias Example for Table Columns

The following SQL statement specifies two aliases, one for the CustomerName column and one for the ContactName column. **Tip:** It require double quotation marks or square brackets if the column name contains spaces:

## Example

SELECT CustomerName AS Customer, ContactName AS [Contact Person]
FROM Customers;

In the following SQL statement we combine four columns (Address, City, PostalCode, and Country) and create an alias named "Address":

SELECT CustomerName, CONCAT(Address,', ',City,', ',PostalCode,', ',Country) AS Address
FROM Customers;

Aliases can be useful when:

* There are more than one table involved in a query
* Functions are used in the query
* Column names are big or not very readable
* Two or more columns are combined together