WHERE

The optional **WHERE** clause is how we tell SQL that we want to limit the results that a query returns. Without a **WHERE** clause, all rows in the table are returned.

A **WHERE** clause consists of a predicate, which is an expression that returns either true or false. SQL will retrieve all the rows in the table for which that predicate returns true.

The Syntax for an SQL query that contains a **WHERE** clause is as follows:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

The following operators can be used in the **WHERE** clause:

Type	Operators
Comparison operators	=, <, >, <>, !=, >=, <= *UnitsInStock > 0*
Logical operators	AND, OR, NOT *UnitsInStock = 0 AND UnitsOnOrder = 0*
"Special" operators	BETWEEN, IN, LIKE *OrderDate BETWEEN '2012/01/01' AND '2012/01/31'*

The AND and OR operators are used to filter records based on more than one condition:

- AND displays a record if all the conditions separated by AND are TRUE.
- OR displays a record if any of the conditions separated by OR is TRUE.
- NOT displays a record if the condition(s) is NOT TRUE.

IN

The IN operator enables us to specify a list of values which we want the expression to match. It is a shorthand way of specifying multiple OR statements that apply to the same column.

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

BETWEEN

Rather than using “greater than or equal to”, “AND” and “less than or equal to”, we can use a BETWEEN expression to specify a range of values. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

Exercise 1: Basic WHERE clauses

Northwind Traders are attempting to learn more about their currently-stocked beverages. They ask you to produce a list of current products that are in that category and that have a high value.

In this exercise, you will use basic WHERE clauses.

Task 1: Create a report that only returns current products

1. Create a new query and save it with a name of “CurrentBeverageProducts.sql”.
2. Write a report that uses the Northwind database and displays the ProductID, ProductName, CategoryID, Discontinued and UnitPrice columns from the dbo.Products table.
3. Execute the query and verify that it returns 77 rows.
4. Add a WHERE clause to the query that asks for only those products with a Discontinued value of 0 (zero).
5. Execute the query and verify that it returns only 69 rows.

Task 2: Write a report that only returns current *products* in category 1

1. Modify your existing report.
2. Add an additional statement to the WHERE clause that further limits the results to products with a CategoryID of 1.
3. Execute the query and verify that it now returns only 11 rows.

Task 3: Write a report that only returns expensive, current products in category 1

1. Modify your existing report.
2. Add an additional statement to the WHERE that further limits the results to products with a unit price greater than or equal to 40.
3. Execute the query and verify that it now retrieves just two products – Cote de Blaye and Ipoh Coffee.

Exercise 2: Using IN and BETWEEN

Northwind tell you they've had some complaints about orders that were placed in a certain month for certain customers. They ask you to produce a report of all the orders that were placed by these customers in that month.

In this exercise, you will use the IN and BETWEEN operators.

Task 1: Create a report that only returns orders for some customers

1. Create a new query and save it with a name of "CustomerComplaints.sql".
2. Write a report that uses the Northwind database and displays the OrderID, CustomerID and OrderDate columns from the dbo.Orders table.
3. Limit the results to only those orders with a CustomerID of either 'ALFKI', 'ERNSH' or 'SIMOB'.
4. Remember that string literals need to be enclosed in 'single quotes'.
5. Execute the query and verify that it returns 43 rows.

Task 2: Write a report that returns orders for some customers in August 1997

1. Modify your existing report.
2. Further limit the results of the query so that it only returns orders with an OrderDate greater than or equal to the 1st August 1997 and an OrderDate of less than or equal to the 31st August 1997.
3. Remember that date literals, like string literals, need to be enclosed in 'single quotes'.
4. Execute the query and verify that there are now just 3 rows – OrderIDs 10633, 10642 and 10643.
5. Note: it is quite likely that you'll get an error at this point! The reason for this being that SQL Server is a US product and the US date format is MDY rather than DMY. There is no 31st month of the year!
6. There are several ways to resolve this issue but the simplest is either to spell out the month, i.e. '31 Aug 1997' or to use YYYY-MM-DD format – '1997-08-31'. Remember to do this for both the start date and the end date otherwise you might be looking for orders between the 8 January and the 31 August!

7. Make sure the query returns the three rows you want: 37 rows and 73 rows are wrong for different reasons

Task 3: Ensure that the query uses IN and BETWEEN

1. Modify your existing report.
2. If it doesn't already, ensure that the report uses IN for the CustomerIDs and BETWEEN for the OrderDates.
3. Notice that the query is much easier to read and doesn't require so many parentheses.
4. Execute the query and verify that it still returns three rows.