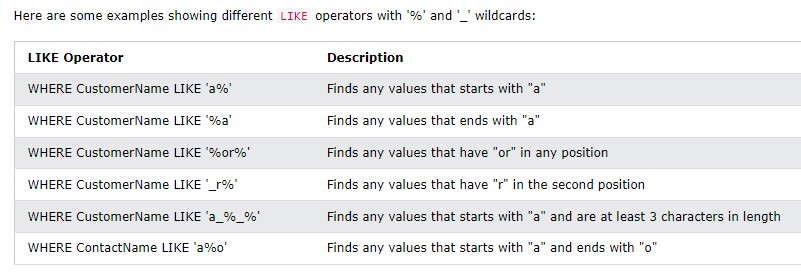
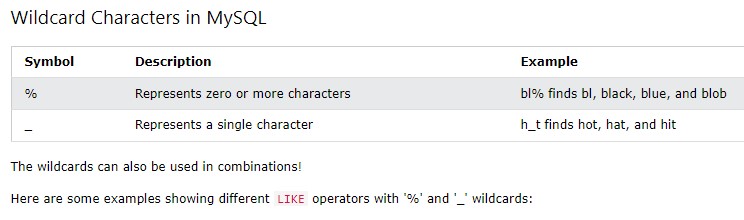
**MySQL Wildcard Characters**

A wildcard character is used to substitute one or more characters in a string.

Wildcard characters are used with the LIKE operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.



**The MySQL IN Operator**

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

The IN operator is a shorthand for multiple OR conditions.

# IN Syntax

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* IN (*value1*, *value2*, ...); or:

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* IN (*SELECT STATEMENT*);

**IN Operator Examples**

The following SQL statement selects all customers that are located in "Germany", "France" or "UK":

SELECT \* FROM Customers

WHERE Country IN ('Germany', 'France', 'UK');

The following SQL statement selects all customers that are NOT located in "Germany", "France" or "UK":

SELECT \* FROM Customers

WHERE Country NOT IN ('Germany', 'France', 'UK');

The following SQL statement selects all customers that are from the same countries as the suppliers:

SELECT \* FROM Customers

WHERE Country IN (SELECT Country FROM Suppliers);

**The MySQL BETWEEN Operator**

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

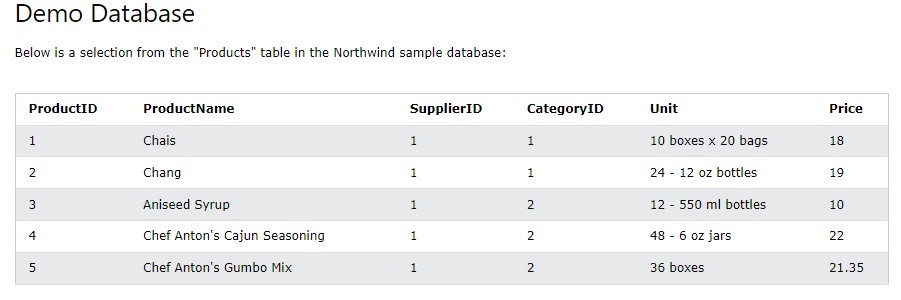
The BETWEEN operator is inclusive: begin and end values are included.

# BETWEEN Syntax

SELECT *column\_name(s)*

FROM *table\_name*

WHERE *column\_name* BETWEEN *value1* AND *value2;*

**BETWEEN Example** 

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 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 with IN Example**

The following SQL statement selects all products with a price between 10 and 20. In addition; do not show products with a CategoryID of 1,2, or 3:

|  |
| --- |
| **Example** |
| SELECT \* FROM Products |

WHERE Price BETWEEN 10 AND 20

AND CategoryID NOT IN (1,2,3);

**BETWEEN Text Values Example**

The following SQL statement selects all products with a ProductName between "Carnarvon Tigers" and "Mozzarella di Giovanni":

|  |
| --- |
| **Example** |
| SELECT \* FROM Products |

WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di

Giovanni'

ORDER BY ProductName;

The following SQL statement selects all products with a ProductName between "Carnarvon Tigers" and "Chef Anton's Cajun Seasoning":

|  |
| --- |
| **Example** |
| SELECT \* FROM Products |

WHERE ProductName BETWEEN "Carnarvon Tigers" AND "Chef Anton's Cajun

Seasoning"

ORDER BY ProductName;

**NOT BETWEEN Text Values Example**

The following SQL statement selects all products with a ProductName not between "Carnarvon Tigers" and "Mozzarella di Giovanni":

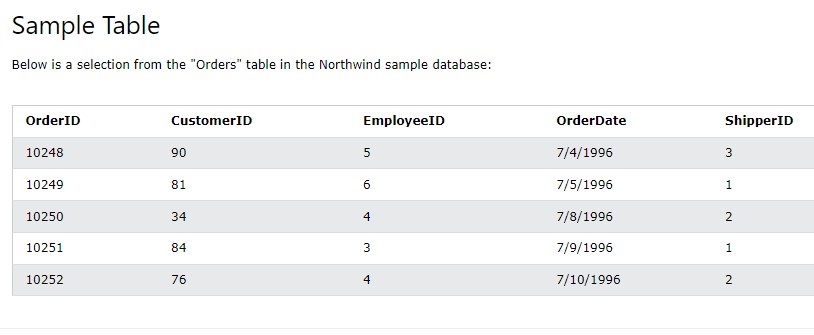
# Example

SELECT \* FROM Products

WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di

Giovanni'

ORDER BY ProductName;



**BETWEEN Dates Example**

The following SQL statement selects all orders with an OrderDate between '01-July-1996' and '31-July-1996':

|  |
| --- |
| **Example** |
| SELECT \* FROM Orders |

WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';

**MySQL Aliases**

Aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

# Alias Column Syntax

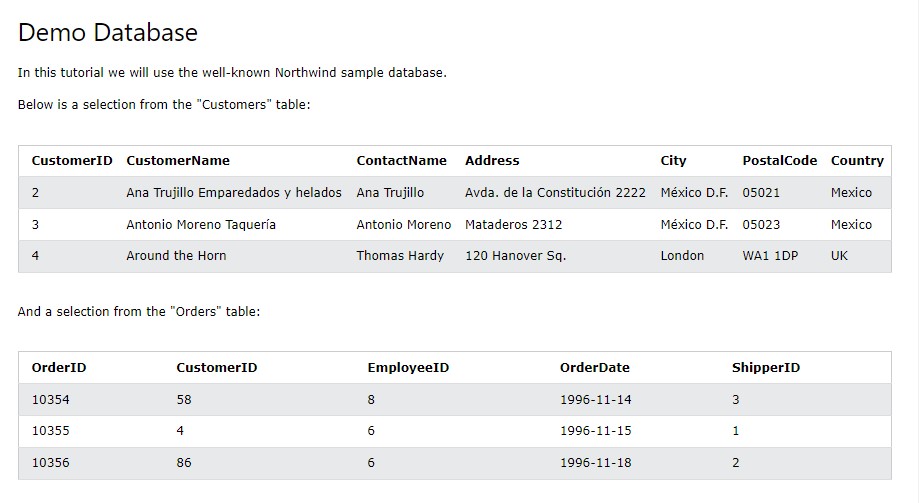
SELECT *column\_name* AS *alias\_name*

FROM *table\_name;*

# Alias Table Syntax

SELECT *column\_name(s)*

FROM *table\_name* AS *alias\_name;*

**Alias for Columns Examples** 

The following SQL statement creates two aliases, one for the CustomerID column and one for the CustomerName column:

|  |
| --- |
| **Example** |
| SELECT CustomerID AS ID, CustomerName AS Customer |

FROM Customers;

The following SQL statement creates two aliases, one for the CustomerName column and one for the ContactName column. **Note:** Single or double quotation marks are required if the alias name contains spaces:

|  |
| --- |
| **Example** |
| SELECT CustomerName AS Customer, ContactName AS "Contact Person" |

FROM Customers;

The following SQL statement creates an alias named "Address" that combine four columns (Address, PostalCode, City and Country):

|  |
| --- |
| **Example** |
| SELECT CustomerName, CONCAT\_WS(', ', Address, PostalCode, City, |

Country) AS Address FROM Customers;

**Alias for Tables Example**

The following SQL statement selects all the orders from the customer with CustomerID=4 (Around the Horn). We use the "Customers" and "Orders" tables, and give them the table aliases of "c" and "o" respectively (Here we use aliases to make the SQL shorter):

|  |
| --- |
| **Example** |
| SELECT o.OrderID, o.OrderDate, c.CustomerName |

FROM Customers AS c, Orders AS o

WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID; The following SQL statement is the same as above, but without aliases:

|  |
| --- |
| **Example** |
| SELECT Orders.OrderID, Orders.OrderDate, Customers.CustomerName |

FROM Customers, Orders

WHERE Customers.CustomerName='Around the Horn' AND Customers.CustomerID=Orders.CustomerID; **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