**The MySQL SELECT DISTINCT**

**Statement**

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

# SELECT DISTINCT Syntax

SELECT DISTINCT *column1*, *column2, ...* FROM *table\_name*;

**SELECT Example Without DISTINCT**

The following SQL statement selects all (including the duplicates) values from the "Country" column in the "Customers" table:

SELECT Country FROM Customers;

**SELECT DISTINCT Examples**

The following SQL statement selects only the DISTINCT values from the "Country" column in the "Customers" table:

SELECT DISTINCT Country FROM Customers;

**The MySQL WHERE Clause**

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

# WHERE Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name* WHERE *condition*;

**note:** The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.!



**WHERE Clause Example**

The following SQL statement selects all the customers from "Mexico":

WHER

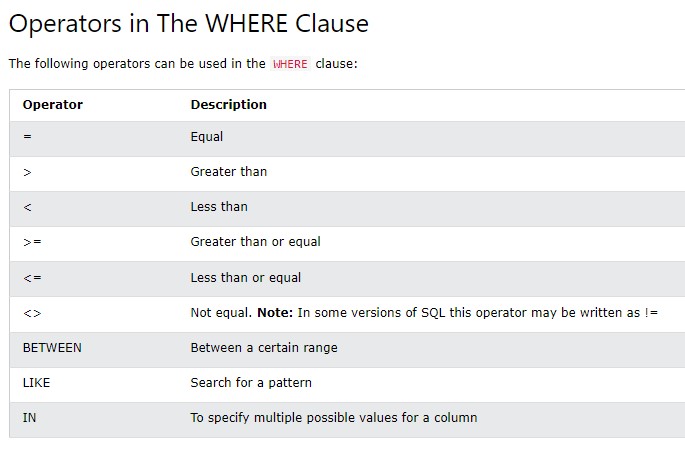
E

Country =

'Mexico

'

;



SELECT \* FROM Customers

**The MySQL AND, OR and NOT Operators**

The WHERE clause can be combined with AND, OR, and NOT operators.

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

* The AND operator displays a record if all the conditions separated by AND are TRUE.
* The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

# AND Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition1* AND *condition2* AND *condition3 ...*;

# OR Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE *condition1* OR *condition2* OR *condition3 ...*;

# NOT Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

WHERE NOT *condition*;

**AND Example**



The following SQL statement selects all fields from "Customers" where country is "Germany" AND city is "Berlin":

# Example

SELECT \* FROM Customers

WHERE Country = 'Germany' AND City = 'Berlin';

**OR Example**

The following SQL statement selects all fields from "Customers" where city is "Berlin" OR "Stuttgart":

|  |
| --- |
| **Example** |
| SELECT \* FROM Customers |

WHERE City = 'Berlin' OR City = 'Stuttgart';

The following SQL statement selects all fields from "Customers" where country is "Germany" OR "Spain":

|  |
| --- |
| **Example** |
| SELECT \* FROM Customers |

WHERE Country = 'Germany' OR Country = 'Spain';

**NOT Example**

The following SQL statement selects all fields from "Customers" where country is NOT "Germany":

# Example

SELECT \* FROM Customers

WHERE NOT Country = 'Germany';

**Combining AND, OR and NOT**

You can also combine the AND, OR and NOT operators.

The following SQL statement selects all fields from "Customers" where country is "Germany" AND city must be "Berlin" OR "Stuttgart" (use parenthesis to form complex expressions):

# Example

SELECT \* FROM Customers

WHERE Country = 'Germany' AND (City = 'Berlin' OR City = 'Stuttgart');

The following SQL statement selects all fields from "Customers" where country is NOT "Germany" and NOT "USA":

|  |
| --- |
| **Example** |
| SELECT \* FROM Customers |

WHERE NOT Country = 'Germany' AND NOT Country = 'USA';

**The MySQL ORDER BY Keyword**

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

# ORDER BY Syntax

SELECT *column1*, *column2, ...*

FROM *table\_name*

ORDER BY *column1, column2, ...* ASC|DESC;



**ORDER BY Example**

The following SQL statement selects all customers from the "Customers" table, sorted by the "Country" column:

|  |
| --- |
| **Example** |
| SELECT \* FROM Customers |

ORDER BY Country;

**ORDER BY DESC Example**

The following SQL statement selects all customers from the "Customers" table, sorted DESCENDING by the "Country" column:

|  |
| --- |
| **Example** |
| SELECT \* FROM Customers |

ORDER BY Country DESC;

**What is a NULL Value?**

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

**Note:** A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

**How to Test for NULL Values?**

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

# IS NULL Syntax

SELECT *column\_names*

FROM *table\_name*

WHERE *column\_name* IS NULL;

# IS NOT NULL Syntax

SELECT *column\_names*

FROM *table\_name*

WHERE *column\_name* IS NOT NULL;



**The IS NULL Operator**

The IS NULL operator is used to test for empty values (NULL values).

The following SQL lists all customers with a NULL value in the "Address" field:

|  |
| --- |
| **Example** |
| SELECT CustomerName, ContactName, Address |

FROM Customers

WHERE Address IS NULL;

Always use IS NULL to look for NULL values.

**The IS NOT NULL Operator**

The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

The following SQL lists all customers with a value in the "Address" field:

# Example

SELECT CustomerName, ContactName, Address

FROM Customers

WHERE Address IS NOT NULL;

**The MySQL UPDATE Statement**

The UPDATE statement is used to modify the existing records in a table.

# UPDATE Syntax

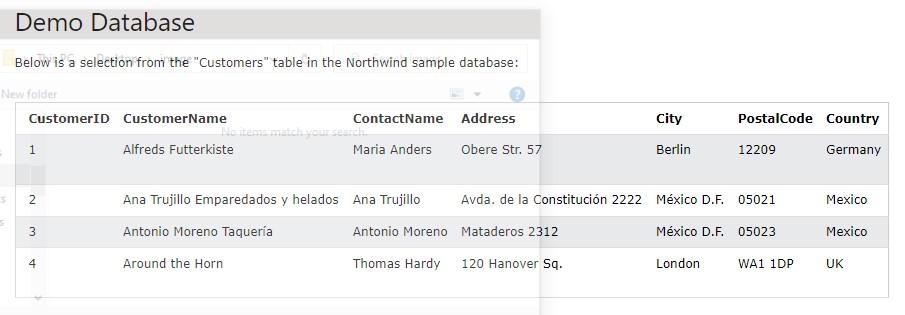
UPDATE *table\_name*

SET *column1* = *value1*, *column2* = *value2*, ... WHERE *condition*;

**Note:** Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

**UPDATE Table**

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.



|  |
| --- |
| **Example** |
| UPDATE Customers |

SET ContactName = 'Alfred Schmidt', City = 'Frankfurt'

WHER

E

CustomerID =

1

;



**UPDATE Multiple Records**

It is the WHERE clause that determines how many records will be updated.

The following SQL statement will update the PostalCode to 00000 for all records where country is "Mexico":

|  |
| --- |
| **Example** |
| UPDATE Customers |

SET PostalCode = 00000

WHERE Country = 'Mexico';

**The MySQL DELETE Statement**



The DELETE statement is used to delete existing records in a table.

# DELETE Syntax

DELETE FROM *table\_name* WHERE *condition*;

**Note:** Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be

deleted. If you omit the

WHER

E

clause, all records in the table will be deleted

!



**SQL DELETE Example**

The following SQL statement deletes the customer "Alfreds Futterkiste" from the "Customers" table:

|  |
| --- |
| **Example** |
| DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste'; |



**Delete All Records**

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM *table\_name*;

The following SQL statement deletes all rows in the "Customers" table, without deleting the table:

|  |
| --- |
| **Example** |
| DELETE FROM Customers; |