## MySQL INSERT INTO Statement

---

The MySQL INSERT INTO Statement

The INSERT INTO statement is used to insert new records in a table.

**INSERT INTO Syntax**

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

---

## MySQL NULL Values

---

**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;
```

---

**MySQL UPDATE Statement**

---

**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;
```

---

**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';
```

---

**MySQL DELETE Statement**

---

**The MySQL DELETE Statement**

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

**DELETE Syntax**

```
DELETE FROM table_name WHERE condition;
```

---

**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 MySQL LIMIT Clause**

The LIMIT clause is used to specify the number of records to return.

The LIMIT clause is useful on large tables with thousands of records. Returning a large number of records can impact performance.

**LIMIT Syntax**

SELECT *column_name(s)*
FROM *table_name*
WHERE *condition*
LIMIT *number*;

**MySQL LIMIT Examples**

**The following SQL statement selects the first three records from the "Customers" table:**

**Example**

```
SELECT * FROM Customers
LIMIT 3;
```

**What if we want to select records 4 - 6 (inclusive)?**

MySQL provides a way to handle this: by using OFFSET.

The SQL query below says "return only 3 records, start on record 4 (OFFSET 3)":

**Example**

```
SELECT * FROM Customers
LIMIT 3 OFFSET 3;
```

---

**ADD a WHERE CLAUSE**

The following SQL statement selects the first three records from the "Customers" table, where the country is "Germany":

**Example**

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

---

**MySQL MIN() and MAX() Functions**

---

**MySQL MIN() and MAX() Functions**

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

**MIN() Syntax**

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;
```

**MAX() Syntax**

```
SELECT MAX(column_name)
FROM table_name
WHERE condition;
```

---

**MIN() Example**

The following SQL statement finds the price of the cheapest product:

**Example**

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

**MAX() Example**

The following SQL statement finds the price of the most expensive product:

**Example**

```
SELECT MAX(Price) AS LargestPrice
FROM Products;
```