**SQL INSERT INTO, NULL Values, Update, Delete, Top, Limit, fetch**

/\*

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, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO table\_name

VALUES (value1, value2, value3, ...);

\*/

-- INSERT INTO Example

-- The following SQL statement inserts a new record in the "Customers" table:

select \* from customers;

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

-- Insert Data Only in Specified Columns

-- It is also possible to only insert data in specific columns.

-- The following SQL statement will insert a new record, but only insert data in the "CustomerName", "City", and "Country" columns (CustomerID will be updated automatically):

INSERT INTO Customers (CustomerName, City, Country)

VALUES ('Cardinal', 'Stavanger', 'Norway');

-- Insert Multiple Rows

-- It is also possible to insert multiple rows in one statement.

-- To insert multiple rows of data, we use the same INSERT INTO statement, but with multiple values:

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES

('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway'),

('Greasy Burger', 'Per Olsen', 'Gateveien 15', 'Sandnes', '4306', 'Norway'),

('Tasty Tee', 'Finn Egan', 'Streetroad 19B', 'Liverpool', 'L1 0AA', 'UK');

select \* from customers;

-----------------------------------------------------------------------

-- SQL NULL Values

-----------------------------------------------------------------------

INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)

VALUES ('', 'Null', ' ', 'Harvey', '4007', 'US');

/\*

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:

SELECT CustomerName, ContactName, Address

FROM Customers

WHERE Address IS NULL;

-- 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:

SELECT CustomerName, ContactName, Address

FROM Customers

WHERE Address IS NOT NULL;

-----------------------------------------------------------------------

-- SQL 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.

UPDATE Customers

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

WHERE CustomerID = 1;

select \* from customers;

-- UPDATE Multiple Records

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

-- The following SQL statement will update the ContactName to "Juan" for all records where country is "Mexico":

select \* from customers where Country='Mexico';

UPDATE Customers

SET ContactName='Juan'

WHERE Country='Mexico';

-- Update Warning!

-- Be careful when updating records. If you omit the WHERE clause, ALL records will be updated!

UPDATE Customers

SET ContactName='Juan';

-----------------------------------------------------------------------

-- SQL DELETE Statement

-----------------------------------------------------------------------

/\*

The SQL 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 WHERE 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:

select \* from Customers WHERE CustomerName='Alfreds Futterkiste';

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; -- Deletes records one by one.

TRUNCATE TABLE table\_name; -- Deletes/removes data from a table at a time.

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

DELETE FROM Customers;

select \* from Customers;

-- Delete a Table

-- To delete the table completely, use the DROP TABLE statement:

-- Remove the Customers table:

DROP TABLE Customers;

-----------------------------------------------------------------------

-- SQL TOP, LIMIT, FETCH FIRST or ROWNUM Clause

-----------------------------------------------------------------------

/\*

The SQL SELECT TOP Clause

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

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

Select only the first 3 records of the Customers table:

Note: Not all database systems support the SELECT TOP clause. MySQL supports the LIMIT clause to select a limited number of records,

while Oracle uses FETCH FIRST n ROWS ONLY and ROWNUM.

\*/

SELECT TOP 3 \* FROM PRACTICE\_DB.SQL.CATEGORIES;

select \* from PRACTICE\_DB.SQL.CATEGORIES LIMIT 5 OFFSET 2;