Introduction to SQL Server INSERT statement

To add one or more rows into a table, you use the INSERT statement. The following illustrates the most basic form of the INSERT statement:

INSERT INTO table\_name (column\_list)

VALUES (value\_list);

Code language: SQL (Structured Query Language) (sql)

Let’s examine this syntax in more detail.

First, you specify the name of the table which you want to insert. Typically, you reference the table name by the schema name e.g., production.products where production is the schema name and products is the table name.

Second, you specify a list of one or more columns in which you want to insert data. You must enclose the column list in parentheses and separate the columns by commas.

If a column of a table does not appear in the column list, SQL Server must be able to provide a value for insertion or the row cannot be inserted.

SQL Server automatically uses the following value for the column that is available in the table but does not appear in the column list of the INSERT statement:

* The next incremental value if the column has an [IDENTITY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-identity/) property.
* The default value if the column has a default value specified.
* The current timestamp value if the data type of the column is a timestamp data type.
* The NULL if the column is nullable.
* The calculated value if the column is a computed column.

Third, you provide a list of values to be inserted in the VALUES clause. Each column in the column list must have a corresponding value in the value list. Also, you must enclose the value list in parentheses.

SQL Server INSERT statement examples

Let’s create a new table named promotions for the demonstration:

CREATE TABLE sales.promotions (

promotion\_id INT PRIMARY KEY IDENTITY (1, 1),

promotion\_name VARCHAR (255) NOT NULL,

discount NUMERIC (3, 2) DEFAULT 0,

start\_date DATE NOT NULL,

expired\_date DATE NOT NULL

);

Code language: SQL (Structured Query Language) (sql)

In this statement, we created a new table named promotions in the sales schema. The promotions table has five columns including promotion identification number, name, discount, start date and expired date.

The promotion identification number is an identity column so its value is automatically populated by the SQL Server when you add a new row to the table.

1) Basic INSERT example

The following statement inserts a new row into the promotions table:

INSERT INTO sales.promotions (

promotion\_name,

discount,

start\_date,

expired\_date

)

VALUES

(

'2018 Summer Promotion',

0.15,

'20180601',

'20180901'

);

Code language: SQL (Structured Query Language) (sql)

In this example, we specified values for four columns in the promotions table. We did not specify a value for the promotion\_id column because SQL Server provides the value for this column automatically.

If the INSERT statement executes successfully, you will get the number of rows inserted. In this case, SQL Server issued the following message:

(1 row affected)

Code language: SQL (Structured Query Language) (sql)

To verify the insert operation, you use the following query:

SELECT

\*

FROM

sales.promotions;

Code language: SQL (Structured Query Language) (sql)

Here is the result as you expected.

SQL Server INSERT example

2) Insert and return inserted values

To capture the inserted values, you use the OUTPUT clause. For example, the following statement inserts a new row into the promotions table and returns the inserted value of the promotion\_id column:

INSERT INTO sales.promotions (

promotion\_name,

discount,

start\_date,

expired\_date

) OUTPUT inserted.promotion\_id

VALUES

(

'2018 Fall Promotion',

0.15,

'20181001',

'20181101'

);

Code language: SQL (Structured Query Language) (sql)

SQL Server INSERT OUTPUT example

To capture inserted values from multiple columns, you specify the columns in the output as shown in the following statement:

INSERT INTO sales.promotions (

promotion\_name,

discount,

start\_date,

expired\_date

) OUTPUT inserted.promotion\_id,

inserted.promotion\_name,

inserted.discount,

inserted.start\_date,

inserted.expired\_date

VALUES

(

'2018 Winter Promotion',

0.2,

'20181201',

'20190101'

);

Code language: SQL (Structured Query Language) (sql)

The following is the output:

SQL Server INSERT OUTPUT multiple columns

3) Insert explicit values into the identity column

Typically, you don’t specify a value for the [identity column](https://www.sqlservertutorial.net/sql-server-basics/sql-server-identity/) because SQL Server will provide the value automatically.

However, in some situations, you may want to insert a value into the identity column such as data migration.

See the following INSERT statement:

INSERT INTO sales.promotions (

promotion\_id,

promotion\_name,

discount,

start\_date,

expired\_date

) OUTPUT inserted.promotion\_id

VALUES

(

4,

'2019 Spring Promotion',

0.25,

'20190201',

'20190301'

);

Code language: SQL (Structured Query Language) (sql)

SQL Server issued the following error:

Cannot insert explicit value for identity column in table 'promotions' when IDENTITY\_INSERT is set to OFF.

Code language: Shell Session (shell)

To insert explicit value for the identity column, you must execute the following statement first:

SET IDENTITY\_INSERT table\_name ON;

Code language: SQL (Structured Query Language) (sql)

To switch the identity insert off, you use the similar statement:

SET IDENTITY\_INSERT table\_name OFF;

Code language: SQL (Structured Query Language) (sql)

Let’s execute the following statements to insert a value for the identity column in the promotions table:

SET IDENTITY\_INSERT sales.promotions ON;

INSERT INTO sales.promotions (

promotion\_id,

promotion\_name,

discount,

start\_date,

expired\_date

)

VALUES

(

4,

'2019 Spring Promotion',

0.25,

'20190201',

'20190301'

);

SET IDENTITY\_INSERT sales.promotions OFF;

Code language: SQL (Structured Query Language) (sql)

In this example, first, we switched the identity insert on, then inserted a row with an explicit value for the identity column, and finally switched the identity insert off.

The following shows the data of the promotions table after the insertion:

SELECT \* FROM sales.promotions;

Code language: SQL (Structured Query Language) (sql)



o add multiple rows to a table at once, you use the following form of the INSERT statement:

INSERT INTO table\_name (column\_list)

VALUES

(value\_list\_1),

(value\_list\_2),

...

(value\_list\_n);

Code language: SQL (Structured Query Language) (sql)

In this syntax, instead of using a single list of values, you use multiple comma-separated lists of values for insertion.

The number of rows that you can insert at a time is 1,000 rows using this form of the INSERT statement. If you want to insert more rows than that, you should consider using multiple INSERT statements, BULK INSERT or a derived table.

Note that this INSERT multiple rows syntax is only supported in SQL Server 2008 or later.

To insert multiple rows returned from a SELECT statement, you use the [INSERT INTO SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert-into-select/) statement.

## SQL Server INSERT multiple rows – examples

We will use the sales.promotions table created in the [previous tutorial](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) for the demonstration.

If you have not yet created the sales.promotions table, you can use the following [CREATE TABLE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) statement:

CREATE TABLE sales.promotions (

promotion\_id INT PRIMARY KEY IDENTITY (1, 1),

promotion\_name VARCHAR (255) NOT NULL,

discount NUMERIC (3, 2) DEFAULT 0,

start\_date DATE NOT NULL,

expired\_date DATE NOT NULL

);

Code language: SQL (Structured Query Language) (sql)

### 1) Inserting multiple rows example

The following statement inserts multiple rows to the sales.promotions table:

INSERT INTO sales.promotions (

promotion\_name,

discount,

start\_date,

expired\_date

)

VALUES

(

'2019 Summer Promotion',

0.15,

'20190601',

'20190901'

),

(

'2019 Fall Promotion',

0.20,

'20191001',

'20191101'

),

(

'2019 Winter Promotion',

0.25,

'20191201',

'20200101'

);

Code language: SQL (Structured Query Language) (sql)

SQL server issued the following message indicating that three rows have been inserted successfully.

(3 rows affected)

Code language: SQL (Structured Query Language) (sql)

Let’s verify the insert by executing the following query:

SELECT

\*

FROM

sales.promotions;

Code language: SQL (Structured Query Language) (sql)

Here is the output:



### 2) Inserting multiple rows and returning the inserted id list example

This example inserts three rows into the sales.promotions table and returns the promotion identity list:

INSERT INTO

sales.promotions (

promotion\_name, discount, start\_date, expired\_date

)

OUTPUT inserted.promotion\_id

VALUES

('2020 Summer Promotion',0.25,'20200601','20200901'),

('2020 Fall Promotion',0.10,'20201001','20201101'),

('2020 Winter Promotion', 0.25,'20201201','20210101');

Code language: SQL (Structured Query Language) (sql)



In this example, we added the OUTPUT clause with the column that we want to return using the inserted.column\_name syntax. If you want to return values from multiple columns, you can use the following syntax:

OUTPUT inserted.column1, inserted.column2...

## Introduction to SQL Server INSERT INTO SELECT statement

To [insert](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) data from other tables into a table, you use the following SQL Server INSERT INTO SELECT statement:

INSERT [ TOP ( expression ) [ PERCENT ] ]

INTO target\_table (column\_list)

query

Code language: SQL (Structured Query Language) (sql)

In this syntax, the statement inserts rows returned by the query into the target\_table.

The query is any valid [SELECT](https://www.sqlservertutorial.net/sql-server-basics/sql-server-select/) statement that retrieves data from other tables. It must return the values that are corresponding to the columns specified in the column\_list.

The TOP clause part is optional. It allows you to specify the number of rows returned by the query to be inserted into the target table. If you use the PERCENT option, the statement will insert the percent of rows instead. Note that it is a best practice to always use the TOP clause with the [ORDER BY](https://www.sqlservertutorial.net/sql-server-basics/sql-server-order-by/) clause.

## SQL Server INSERT INTO SELECT examples

Let’s [create a table](https://www.sqlservertutorial.net/sql-server-basics/sql-server-create-table/) named addresses for the demonstration:

CREATE TABLE sales.addresses (

address\_id INT IDENTITY PRIMARY KEY,

street VARCHAR (255) NOT NULL,

city VARCHAR (50),

state VARCHAR (25),

zip\_code VARCHAR (5)

);

Code language: SQL (Structured Query Language) (sql)

### 1) Insert all rows from another table example

The following statement [inserts](https://www.sqlservertutorial.net/sql-server-basics/sql-server-insert/) all addresses from the customers table into the addresses table:

INSERT INTO sales.addresses (street, city, state, zip\_code)

SELECT

street,

city,

state,

zip\_code

FROM

sales.customers

ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)

To verify the insert, you use the following query:

SELECT

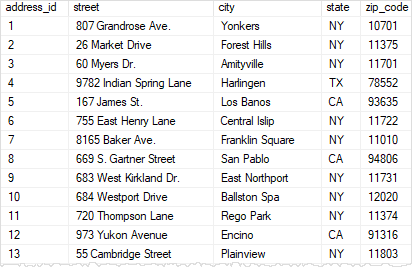
\*

FROM

sales.addresses;

Code language: SQL (Structured Query Language) (sql)

Here is the result:



### 2) Insert some rows from another table example

Sometimes, you just need to insert some rows from another table into a table. In this case, you limit the number of rows returned from the query by using conditions in the [WHERE](https://www.sqlservertutorial.net/sql-server-basics/sql-server-where/) clause.

The following statement adds the addresses of the stores located in Santa Cruz and Baldwin to the addresses table:

INSERT INTO

sales.addresses (street, city, state, zip\_code)

SELECT

street,

city,

state,

zip\_code

FROM

sales.stores

WHERE

city IN ('Santa Cruz', 'Baldwin')

Code language: SQL (Structured Query Language) (sql)

SQL Server returned the following message indicating that two rows have been inserted successfully.

(2 rows affected)

Code language: SQL (Structured Query Language) (sql)

### 3) Insert the top N of rows

First, you use the following statement to delete all rows from the addresses table:

TRUNCATE TABLE sales.addresses;

Code language: SQL (Structured Query Language) (sql)

Second, to insert the top 10 customers sorted by their first names and last names, you use the INSERT TOP INTO SELECT statement as follows:

INSERT TOP (10)

INTO sales.addresses (street, city, state, zip\_code)

SELECT

street,

city,

state,

zip\_code

FROM

sales.customers

ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)

SQL Server returned the following message showing that ten rows have been inserted successfully.

(10 rows affected)

Code language: SQL (Structured Query Language) (sql)

### 4) Insert the top percent of rows

Instead of using an absolute number of rows, you can insert a percent number of rows into a table.

First, [truncate](https://www.sqlservertutorial.net/sql-server-basics/sql-server-truncate-table/) all rows from the addresses table:

TRUNCATE TABLE sales.addresses;

Code language: SQL (Structured Query Language) (sql)

Second, insert top two percent of rows from the customers table sorted by first names and last names into the addresses table:

INSERT TOP (10) PERCENT

INTO sales.addresses (street, city, state, zip\_code)

SELECT

street,

city,

state,

zip\_code

FROM

sales.customers

ORDER BY

first\_name,

last\_name;

Code language: SQL (Structured Query Language) (sql)

SQL Server issued the following message indicating that 145 rows have been inserted successfully.

(145 rows affected)