LAB - Insert

In this lab, you will learn how to use the `INSERT` statement to add a new row to a table.

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);
```

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` 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.

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
);
```

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'
    );
```

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)
```

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

```
SELECT
    *
FROM
    sales.promotions;
```

Here is the result as you expected.

promotion_id	promotion_name	discount	start_date	expired_date
1	2018 Summer Promotion	0.15	2018-06-01	2018-09-01

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'
    );
```

promotion_id
2

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'
    );
```

The following is the output:

promotion_id	promotion_name	discount	start_date	expired_date
3	2018 Winter Promotion	0.15	2018-12-01	2019-01-01

3) Insert explicit values into the identity column

Typically, you don't specify a value for the identity column 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'
    );
```

SQL Server issued the following error:

```
Cannot insert explicit value for identity column in table 'promotions' when
IDENTITY_INSERT is set to OFF.
```

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

```
SET IDENTITY_INSERT table_name ON;
```

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

```
SET IDENTITY_INSERT table_name OFF;
```

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;
```

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;
```

promotion_id	promotion_name	discount	start_date	expired_date
1	2018 Summer Promotion	0.15	2018-06-01	2018-09-01
2	2018 Fall Promotion	0.15	2018-10-01	2018-11-01
3	2018 Winter Promotion	0.15	2018-12-01	2019-01-01
4	2019 Spring Promotion	0.25	2019-02-01	2019-03-01

In this lab, you have learned how to use the SQL Server `INSERT` statement to add a new row to a table.