LAB - Drop Table

In this lab, you will learn how to use the SQL Server `DROP TABLE` statement to remove one or more tables from a database.

Sometimes, you want to remove a table that is no longer in use. To do this, you use the following `DROP TABLE` statement:

```
DROP TABLE [IF EXISTS]  [database_name.][schema_name.]table_name;
```

In this syntax:

- First, specify the name of the table to be removed.
- Second, specify the name of the database in which the table was created and the name of the schema to which the table belongs. The database name is optional. If you skip it, the `DROP TABLE` statement will drop the table in the currently connected database.
- Third, use `IF EXISTS` clause to remove the table only if it exists. The `IF EXISTS` clause has been supported since SQL Server 2016 13.x. If you remove a table that does not exist, you will get an error. The `IF EXISTS` clause conditionally removes the table if it already exists.

When SQL Server drops a table, it also deletes all data, triggers, constraints, permissions of that table. Moreover, SQL Server does not explicitly drop the views and stored procedures that reference the dropped table. Therefore, to explicitly drop these dependent objects, you must use the `DROP VIEW` and `DROP PROCEDURE` statement.

SQL Server allows you to remove multiple tables at once using a single `DROP TABLE` statement as follows:

```
DROP TABLE [database_name.][schema_name.]table_name_1,
           [database_name.][schema_name.]table_name_2,
           ...
           [database_name.][schema_name.]table_name_n;
```

SQL Server `DROP TABLE` examples

Let's see some examples of using the SQL Server `DROP TABLE` statement.

A) Drop a table that does not exist

The following statement removes a table named `revenues` in the `sales` schema:

```
DROP TABLE IF EXISTS sales.revenues;
```

In this example, the `revenues` table does not exist. Because it uses the `IF EXISTS` clause, the statement executes successfully with no table deleted.

B) Drop a single table example

The following statement creates a new table named `delivery` in the `sales` schema:

```
CREATE TABLE sales.delivery (
    delivery_id INT PRIMARY KEY,
    delivery_note VARCHAR (255) NOT NULL,
    delivery_date DATE NOT NULL
);
```

To remove the `delivery` table, you use the following statement:

```
DROP TABLE sales.delivery;
```

C) Drop a table with a foreign key constraint example

The following statement creates two new tables named `supplier_groups` and `suppliers` in the `procurement` schema:

```
CREATE SCHEMA procurement;
GO

CREATE TABLE procurement.supplier_groups (
    group_id INT IDENTITY PRIMARY KEY,
    group_name VARCHAR (50) NOT NULL
);

CREATE TABLE procurement.suppliers (
    supplier_id INT IDENTITY PRIMARY KEY,
    supplier_name VARCHAR (50) NOT NULL,
    group_id INT NOT NULL,
    FOREIGN KEY (group_id) REFERENCES procurement.supplier_groups (group_id)
);
```

Let's try to drop the `supplier_groups` table:

```
DROP TABLE procurement.supplier_groups;
```

SQL Server issued the following error:

```
Could not drop object 'procurement.supplier_groups' because it is referenced by a
FOREIGN KEY constraint.
```

SQL Server does not allow you to delete a table that is referenced by a foreign constraint. To delete this table, you must drop the referencing foreign key constraint or referencing table first. In this case, you have to drop the foreign key constraint in the `suppliers` table or the `suppliers` table first before removing the `supplier_groups` table.

```
DROP TABLE procurement.supplier_groups;
DROP TABLE procurement.suppliers;
```

If you use a single `DROP TABLE` statement to remove both tables, the referencing table must be listed first as shown in the query below:

```
DROP TABLE procurement.suppliers, procurement.supplier_groups;
```

In this lab, you have learned how to use the SQL Server `DROP TABLE` statement to remove one or more tables from a database.