LAB - Create Table

in this lab, you will learn how to use the SQL Server `CREATE TABLE` statement to create a new table.

Tables are used to store data in the database. Tables are uniquely named within a database and schema. Each table contains one or more columns. And each column has an associated data type that defines the kind of data it can store e.g., numbers, strings, or temporal data.

To create a new table, you use the `CREATE TABLE` statement as follows:

```
CREATE TABLE [database_name.][schema_name.]table_name (
    pk_column data_type PRIMARY KEY,
    column_1 data_type NOT NULL,
    column_2 data_type,
    ...,
    table_constraints
);
```

In this syntax:

- First, specify the name of the database in which the table is created. The `database_name` must be the name of an existing database. If you don't specify it, the `database_name` defaults to the current database.
- Second, specify the schema to which the new table belongs.
- Third, specify the name of the new table.
- Fourth, each table should have a primary key which consists of one or more columns. Typically, you list the primary key columns first and then other columns. If the primary key contains only one column, you can use the `PRIMARY KEY` keywords after the column name. If the primary key consists of two or more columns, you need to specify the `PRIMARY KEY` constraint as a table constraint. Each column has an associated data type specified after its name in the statement. A column may have one or more column constraints such as `NOT NULL` and `UNIQUE`.
- Fifth, a table may have some constraints specified in the table constraints section such as `FOREIGN KEY`, `PRIMARY KEY`, `UNIQUE` and `CHECK`.

Note that `CREATE TABLE` is complex and has more options than the syntax above. We will gradually introduce you to each individual options in the subsequent tutorials.

Example

The following statement creates a new table named `sales.visits` to track the customer in-store visits:

```
CREATE TABLE sales.visits (
    visit_id INT PRIMARY KEY IDENTITY (1, 1),
    first_name VARCHAR (50) NOT NULL,
    last_name VARCHAR (50) NOT NULL,
    visited_at DATETIME,
    phone VARCHAR(20),
    store_id INT NOT NULL,
    FOREIGN KEY (store_id) REFERENCES sales.stores (store_id)
);
```

In this example:

Because we do not specify the name of the database explicitly in which the table is created, the visits table is created in the `BikeStores` database. For the schema, we specify it explicitly, therefore, the visits table is created in the sales schema.

The `visits` table contains six columns:

- The `visit_id` column is the primary key column of the table. The `IDENTITY(1,1)` instructs SQL Server to automatically generate integer numbers for the column starting from one and increasing by one for each new row.
- The `first_name` and `last_name` columns are character string columns with `VARCHAR` type. These columns can store up to 50 characters.
- The `visited_at` is a `DATETIME` column that records the date and time at which the customer visits the store.
- The `phone` column is a varying character string column which accepts `NULL`.
- The `store_id` column stores the identification numbers which identify the store where the customer visited.
- At the end of the table's definition is a `FOREIGN KEY` constraint. This foreign key ensures that the values in the `store_id` column of the `visits` table must be available in the `store_id` column in the `stores` table. You will learn more about the `FOREIGN KEY` constraint in the next tutorial.

In this lab, you have learned how to use the `CREATE TABLE` statement to create a new table in a database.