CREATE, DROP, and INSERT Table

In this lesson, we will take a look at three commands regarding relations/tables.

CREATE TABLE

Creating a basic table involves naming the table and defining its columns and the data type for each column.

The SQL **CREATE TABLE** statement is used to create a new table.

Syntax

The basic syntax of the `CREATE TABLE` statement is as follows:

```
CREATE TABLE table_name(

   column1 datatype,

   column2 datatype,

   column3 datatype,
```

```
   columns datatype,

   .....

   columnN datatype,

   PRIMARY KEY(one or more columns)

);
```

`CREATE TABLE` is the keyword telling the database system what you want to do. The unique name or identifier for the table follows the `CREATE TABLE` statement.

Then, in brackets, comes the list defining each column in the table and what data type it is.

Example

The following code block is an example which creates a CUSTOMERS table with ID as a primary key and **NOT NULL** is the constraint showing that these fields cannot be `NULL` while creating records in this table:

```
CREATE TABLE CUSTOMERS(

   ID       INT                    NOT NULL,

   NAME    VARCHAR (20)         NOT NULL,

   AGE      INT                    NOT NULL,

   ADDRESS    CHAR (25),

   SALARY     DECIMAL (18, 2),

   PRIMARY KEY   (ID)

);
```

You can verify that your table has been created successfully by using the **DESC command**:

```
CREATE TABLE CUSTOMERS(
```

```
    ID      INT                 NOT NULL,
    NAME VARCHAR (20)           NOT NULL,
    AGE    INT                  NOT NULL,

    ADDRESS    CHAR (25) ,
    SALARY     DECIMAL (18, 2),  /* The (18,2) simply means that we can have 18 digits with 2 of
    PRIMARY KEY (ID)
);

DESC CUSTOMERS;
```

The `DESC` command in **line 10** creates a table that contains information regarding the columns: name, data type, constraints.

Now, you have a CUSTOMERS table available in your database which you can use to store the required information related to customers.

DROP TABLE

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

> You should be very careful while using this command because once a table is deleted, all the information available in that table will also be lost forever.

Syntax

The basic syntax of the `DROP TABLE` statement is as follows:

```
DROP TABLE table_name;
```

Example

Now let us delete the CUSTOMERS table we created above:

```
CREATE TABLE CUSTOMERS(
   ID    INT               NOT NULL,
   NAME VARCHAR (20)       NOT NULL,
   AGE   INT               NOT NULL,
   ADDRESS   CHAR (25) ,
   SALARY    DECIMAL (18, 2),  /* The (18,2) simply means that we can have 18 digits with 2 of
```
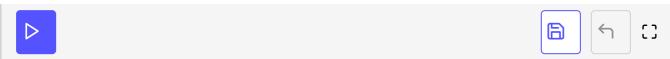
```
    PRIMARY KEY (ID)
);

DESC CUSTOMERS;

DROP TABLE CUSTOMERS;

DESC CUSTOMERS;
```

Now, if we try using the `DESC` command, we will get the above error which simply states that there is no table called CUSTOMERS in our ri_db database. Thus we have successfully dropped/deleted the CUSTOMERS table.

INSERT INTO

The SQL **INSERT INTO** statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the `INSERT INTO` statement which are shown below.

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)

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

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column name(s) in the SQL query if you are adding values for all the columns in the table. But make sure the order of the values is the same as the columns in the table.

The SQL `INSERT INTO` syntax will be as follows:

```
INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN);
```

Example

The following statements would create six records in the CUSTOMERS table:

```sql
CREATE TABLE CUSTOMERS(
   ID    INT              NOT NULL,
   NAME VARCHAR (20)      NOT NULL,
   AGE   INT              NOT NULL,
   ADDRESS   CHAR (25) ,
   SALARY    DECIMAL (18, 2),  /* The (18,2) simply means that we can have 18 digits with 2 of
   PRIMARY KEY (ID)
);

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (1, 'Mark', 32, 'Texas', 50000.00 );

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (2, 'John', 25, 'NY', 65000.00 );

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (3, 'Emily', 23, 'Ohio', 20000.00 );

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (4, 'Bill', 25, 'Chicago', 75000.00 );

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (5, 'Tom', 27, 'Washington', 35000.00 );

INSERT INTO CUSTOMERS (ID, NAME, AGE, ADDRESS, SALARY)
VALUES (6, 'Jane', 22, 'Texas', 45000.00 );
```

The code written from **lines 10-26** will create the following table:

ID	NAME	AGE	ADDRESS	SALARY
1	Mark	32	Texas	50000.00
2	John	25	NY	65000.00
3	Emily	23	Ohio	20000.00
4	Bill	25	Chicago	75000.00
5	Tom	27	Washington	35000.00

| 6 | Jane | 22 | Texas | 45000.00 |

Quick quiz!

Is the following INSERT INTO statement correct?

```
INSERT INTO CUSTOMERS
VALUES (7, 'Troy', 'LA', 40000.00 );
```

COMPLETED 0%

1 of 1

In the next lesson, we will see how the `SELECT` statement is used to retrieve the columns of a table.