## Introduction to DDL Commands

**CREATE TABLE**

**Introduction**

In this lesson, we will learn how to create a new table in the database. When creating a table, we use CREATE TABLE statement. Here is the syntax:

CREATE TABLE database\_name.table\_name (

column1 data\_type,

column2 data\_type,

...

);

A CREATE TABLE statement specifies the following attributes of the new table:

* The name of the new table
* The database in which the new table is created. This is optional. You don't have to specify. For instance, we could have typed CREATE TABLE table\_name (...);
* The name of the columns
* The declared data type of each column

Now, let's create a table in the *students*database. The *students* database has only one table named *student\_table*. If you type the .table command in the playground, the table name(s) of the database will return. .table command is special to SQLite.

query:

.table

output:

student\_table

Alright, we'll now add a table to the *students* database in the next part.

**Example**

We will add a table named *teachers*to our students database.Each teacher has the following information:

* Name
* Lesson
* Salary

Let's create the table.

query:

CREATE TABLE teachers (

name TEXT,

lesson TEXT,

salary INT

);

Now, display the tables in the students database. We must have two tables: student\_table, teachers. query:

.table

output:

student\_table teachers

**⚠️Avoid:**

* Don't try to run DDL commands on the playground. Due to some technical limitations, DDL commands may not work properly. However, you can run DDL commands using DB Browser for SQLite program installed on your local machines. You may also use other RDBMS.

### Constraints

Constraints are the rules specified for data in a table. We can limit the type of data that will go into a table with the constraints. We can define the constraints with the CREATE TABLE statement or ALTER TABLE statement.

Syntax

CREATE TABLE table\_name (

column1 DATATYPE CONSTRAINT,

column1 DATATYPE CONSTRAINT,

...

);

The following are the most commonly used constraints in SQL:

**Constraints**

| **Constraint Name** | **Definition** |
| --- | --- |
| NOT NULL | Ensures that a column cannot have a NULL value |
| DEFAULT | Sets a default value for a column when no value is specified |
| UNIQUE | Ensures that all values in a column are different |
| PRIMARY KEY | Uniquely identifies each row in a table |
| FOREIGN KEY | Uniquely identifies a row/record in another table |

We will start with the PRIMARY KEY constraint.

Q: What are Constraints in SQL?  
A: **NOT NULL** - Restricts NULL value from being inserted into a column. **DEFAULT** - Automatically assigns a default value if no value has been specified for the field. **UNIQUE** - Ensures unique values to be inserted into the field. **PRIMARY KEY** - Uniquely identifies each record in a table. **FOREIGN KEY** - Ensures referential integrity for a record in another table.

### Primary Key

The primary key is a column in our table that makes each row (aka, record) unique.

**💡 Tips:** The primary key is used to uniquely identify each record

A table can have only one primary key. The primary key can consist of one or multiple columns. We define the PRIMARY KEY in the CREATE TABLE statement.  
  
Here is the syntax:

CREATE TABLE table\_name(

column\_1 INT PRIMARY KEY,

column\_2 TEXT,

...

);

It's also possible to define the primary key at the end of the statement. It's very simple syntax: we just say PRIMARY KEY and put the column name inside the parentheses.

CREATE TABLE table\_name(

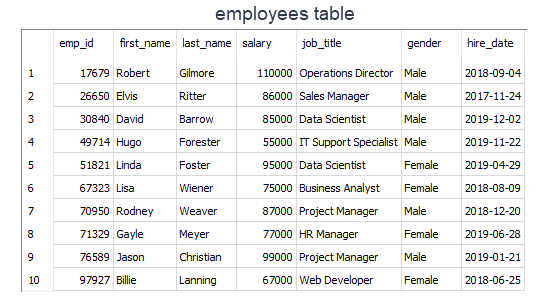
column\_1 INT,

column\_2 TEXT,

...

PRIMARY KEY (column\_1)

);

We define the primary key to uniquely identify each record. That means the value in the primary key column cannot be repeated. Thus we should choose an appropriate column that meets the criteria of the primary key.  
Let's take the example of employees table. Which column do you think as the best fit for the primary key?   
The answer is emp\_id. Because it has no repetition of data and uniquely identifies each record. In this table, each record represents an employee.  


The primary key column cannot contain NULL values. You don't get an error when you don't define a primary key in your table. However, good database design starts with a primary key. In fact, our tables in the database need a primary key. That will ensure row-level accessibility. With the primary key, we can query each table row individually.

Suppose that we don't have any tables yet in our company database. Let's create the employees table from scratch and define a primary key.

CREATE TABLE employees(

emp\_id INT NOT NULL PRIMARY KEY,

first\_name TEXT,

last\_name TEXT,

salary INT,

job\_title TEXT,

gender TEXT,

hire\_date TEXT);

You may notice that we use NOT NULL constraint with the primary key. But it's not mandatory to use it with the primary key since the primary key is implicitly NOT NULL.   
  
Q: What is a Primary Key?  
A: The PRIMARY KEY constraint uniquely identifies each row in a table. It must contain UNIQUE values and has an implicit NOT NULL constraint.

**Foreign Key**

Foreign key is a column in a table that uniquely identifies each row of another table. That column refers to a primary key of another table. This creates a kind of link between the tables.

Assume that we have a database called *store*. Let's create two tables as customers and orders inside the *store*database.

query:

CREATE TABLE customers (customer\_id INT PRIMARY KEY,

first\_name TEXT,

second\_name TEXT);

Now, we will create the second table called *orders.* In the orders table, we defined the FOREIGN KEY for the customer\_id column and pointed it using REFERENCES to the customer\_id column in the *customers* table.  
  
query:

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

order\_number INT,

customer\_id INT,

FOREIGN KEY (customer\_id)

REFERENCES customers (customer\_id)

);

* The customer\_id column in the customers table is the PRIMARY KEY in the customers table.
* The customer\_id column in the orders table is the FOREIGN KEY in the orders table.
* The customers table is called a parent table.
* The orders table is called a child table.

Let's insert some values into our newly created tables. We use INSERT INTO statement to add data. The syntax is very straightforward.

INSERT INTO table\_name (column1, column2)

VALUES (value1, value2);

We start to enter values to the customers table first.

INSERT INTO customers (customer\_id, first\_name, last\_name)

VALUES (1, 'Robert', 'Cursor'),

(2, 'Julia', 'Loyds'),

(3, 'Jack', 'Morgten');

Now, let's do it for the orders table.

INSERT INTO orders (order\_id, order\_number, customer\_id)

VALUES (1, 101, 3),

(2, 102, 3),

(3, 103, 2),

(4, 104, 1);

But, what did we achieved by defining FOREIGN KEY? Using foreign keys ensures that not to destroy the links between the tables. Besides, it prevents invalid data entry into the foreign key column. For instance, you cannot add a new customer\_id value to the orders table.

query:

INSERT INTO orders (order\_id, order\_number, customer\_id)

VALUES (5, 105, 4);

If you run this query, you get a "FOREIGN KEY constraint failed" error. This is also called referential integrity.  Since there is no customer\_id '4' in the customers table, we cannot create a new customer\_id in the orders table.  
 

Q:What is a Foreign Key?  
A: A FOREIGN KEY comprises of single or collection of fields in a table that essentially refer to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables. The table with the foreign key constraint is labelled as the child table, and the table containing the candidate key is labelled as the referenced or parent table.

### Not Null

A column can include NULL values. A NULL value is a special value that means the value is unknown or does not exist. We can specify if a column can hold NULL values or not when we create a table. By default, all columns (except primary key's column) in a table can hold NULL values unless we explicitly specify NOT NULL constraints.

We define NOT NULL constraint for a column using the following syntax:

CREATE TABLE table\_name (

column\_name type\_name NOT NULL,

...);

The NOT NULL constraint follows the type of the column as you see in the syntax.  
Let's give an example. Suppose that order\_number column cannot hold NULL values. That means every order should have an order number.

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

order\_number INT NOT NULL,

customer\_id INT,

FOREIGN KEY (customer\_id)

REFERENCES customers (customer\_id)

);

Let's try to insert a null value to the customer\_id.

INSERT INTO orders (order\_number)

VALUES (NULL);

### If you run the query, you will get a "NOT NULL constraint failed" error.