

SQL Default

SQL Additional Topics

Related Topics

SQL PRIMARY KEY

SQL CREATE INDEX

SQL FOREIGN KEY

SQL UNIQUE Constraint

SQL NOT NULL Constraint

SQL DEFAULT Constraint

SQL Create Index

SQL Constraints

Examples - Q Search tutorials and examples

In this tutorial, we'll learn about constraints in SQL with the help of examples.

In a database table, we can add rules to a column known as **constraints**. These rules control the data that can be stored in a column.

∷≡

For example, if a column has NOT NULL constraint, it means the column cannot store NULL values.

The constraints used in SQL are:



Note: These constraints are also called integrity constraints.

NOT NULL Constraint

The NOT NULL constraint in a column means that the column cannot store NULL values. For example,

```
CREATE TABLE Colleges (
college_id INT NOT NULL,
college_code VARCHAR(20) NOT NULL,
college_name VARCHAR(50)
                                                                        Run Code »
```

Here, the college_id and the college_code columns of the Colleges table won't allow NULL values.

To learn more, visit SQL NOT NULL Constraint.

UNIQUE Constraint

The UNIQUE constraint in a column means that the column must have unique value. For example,

```
CREATE TABLE Colleges (
college_id INT NOT NULL UNIQUE,
college_code VARCHAR(20) UNIQUE,
college_name VARCHAR(50)
                                                                         Run Code >>>
```

Here, the value of the college_code column must be unique. Similarly, the value of college_id must be unique as well as it cannot store NULL values.

To learn more, visit SQL UNIQUE Constraint.

PRIMARY KEY Constraint

The PRIMARY KEY constraint is simply a combination of NOT NULL and UNIQUE constraints. It means that the column value is used to uniquely identify the row. For example,

```
CREATE TABLE Colleges (
college_id INT PRIMARY KEY,
college_code VARCHAR(20) NOT NULL,
college_name VARCHAR(50)
                                                                        Run Code >>
```

Here, the value of the college_id column is a unique identifier for a row. Similarly, it cannot store NULL value and must be UNIQUE.

To learn more, visit SQL PRIMARY KEY.

FOREIGN KEY Constraint

The FOREIGN KEY (REFERENCES in some databases) constraint in a column is used to reference a record that exists in another table. For example,

```
CREATE TABLE Orders (
order_id INT PRIMARY KEY,
customer_id int REFERENCES Customers(id)
                                                                        Run Code »
```

Here, the value of the college_code column references the row in another table named Customers.

It means that the value of <code>customer_id</code> in the <code>Orders</code> table must be a value from the <code>id</code> column of the Customers table.

To learn more, visit SQL FOREIGN KEY.

CHECK Constraint

The CHECK constraint checks the condition before allowing values in a table. For example,

```
CREATE TABLE Orders (
order_id INT PRIMARY KEY,
amount int CHECK (amount >= 100)
                                                                        Run Code »
```

Here, the value of the amount column must be greater than or equal to 100. If not, the SQL statement results in an error.

To learn more, visit SQL CHECK Constraint.

DEFAULT Constraint

The DEFAULT constraint is used to set the default value if we try to store NULL in a column. For example,

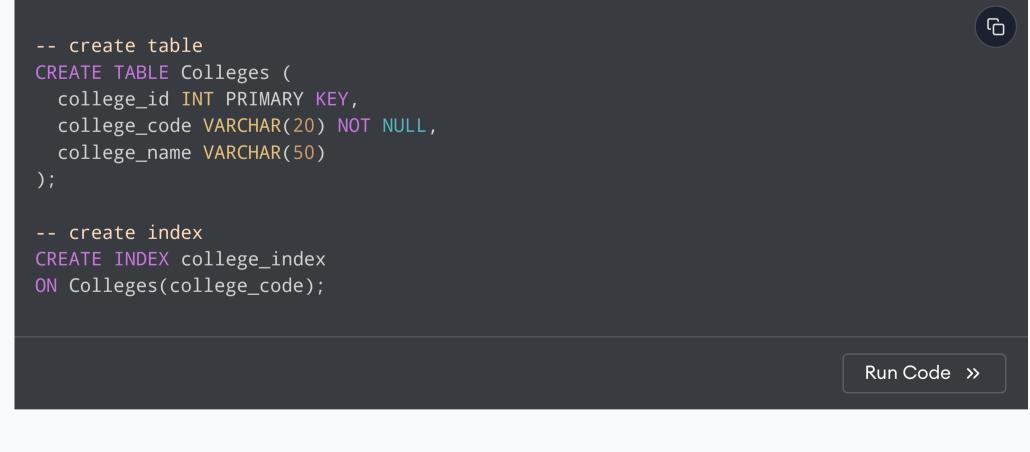
```
CREATE TABLE College (
college_id INT PRIMARY KEY,
college_code VARCHAR(20),
college_country VARCHAR(20) DEFAULT 'US'
                                                                         Run Code >>
```

Here, the default value of the college_country column is US.

If we try to store the NULL value in the college_country column, its value will be US. To learn more, visit SQL DEFAULT Constraint.

CREATE INDEX Constraint

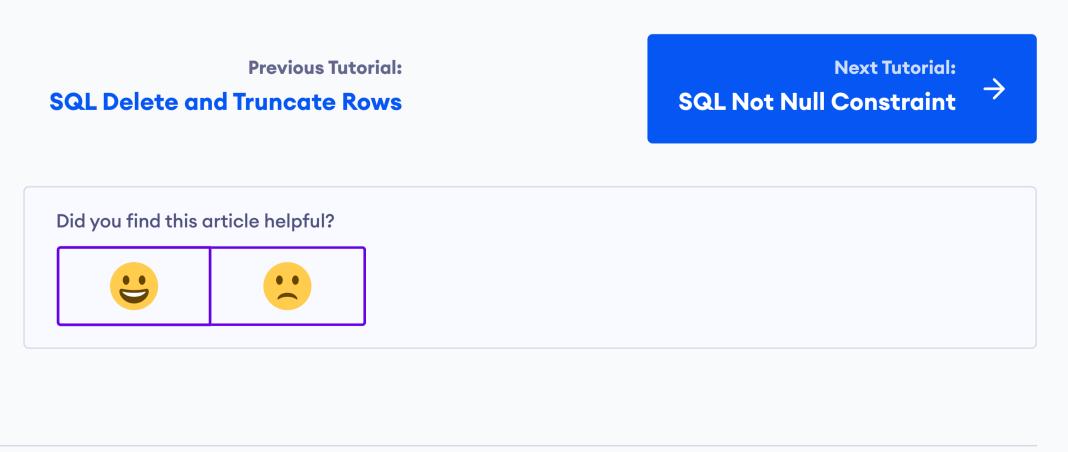
If a column has CREATE INDEX constraint, it's faster to retrieve data if we use that column for data retrieval. For example,



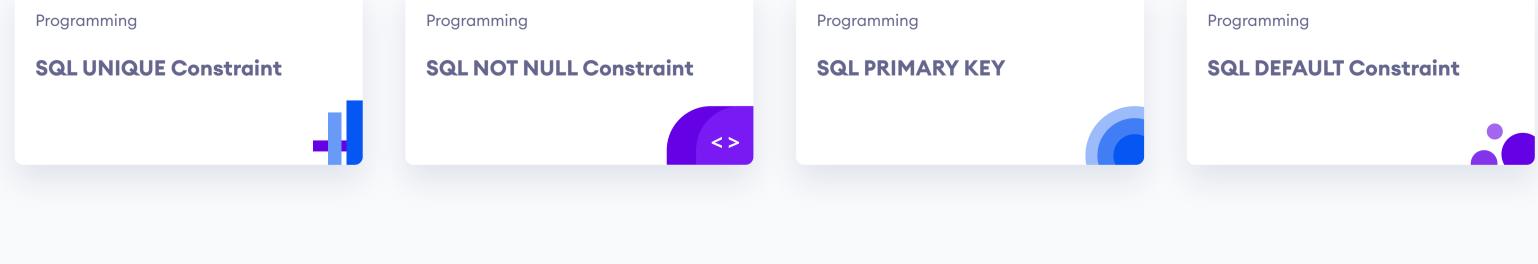
Here, the SQL command creates an index named [customers_index] on the [Customers] table using customer_id column.

Note: We cannot see the speed difference with less records in a table. However, we can easily notice the speed difference between using indexes and not using indexes.

To learn more, visit SQL CREATE INDEX.



Related Tutorials Programming







DSA Tutorial