SQL Introduction
SQL SELECT (I)
SQL SELECT (II)
SQL JOIN
SQL DATABASE & TABLE
SQL Insert, Update and Delete
SQL Constraints
- SQL Constraints
- SQL Not Null Constraint
- SQL Unique Constraints
- SQL Primary Key
- SQL Foreign Key
- SQL Check
- SQL Default
- SQL Create Index

SQL Additional Topics

SQL UNIQUE Constraint

In this tutorial, we'll learn to use the UNIQUE constraint with the help of examples.

In SQL, the `UNIQUE` constraint in a column means that the column must have unique values. For example,

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

Run Code »

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

UNIQUE Vs DISTINCT

The `UNIQUE` constraint is used to make column's value unique. However, to select unique rows from the table, we have to use SQL SELECT DISTINCT. For example,

```
SELECT DISTINCT country
FROM Customers;
```

Run Code »

Here, the SQL command selects unique countries from the Customers table.

Count UNIQUE Rows

If we need to count the number of unique rows, we can use the COUNT() function with the SELECT DISTINCT clause. For example,

```
SELECT COUNT(DISTINCT country)
FROM Customers;
```

Run Code »

Here, the SQL command returns the count of unique countries.

Related Topics
SQL CREATE INDEX
SQL PRIMARY KEY
SQL Constraints
SQL NOT NULL Constraint
SQL DEFAULT Constraint
SQL SELECT DISTINCT Statement

UNIQUE Constraint With Alter Table

We can also add the `UNIQUE` constraint to an existing column using the ALTER TABLE command. For example,

For single column

```
ALTER TABLE Colleges
ADD UNIQUE (college_id);
```

For multiple column

```
ALTER TABLE Colleges
ADD UNIQUE UniqueCollege (college_id, college_code);
```

Here, the SQL command adds the `UNIQUE` constraint to the specified column(s) in an existing table.

Error When Inserting Duplicate Values

If we try to insert duplicate values in a column with the `UNIQUE` constraint, we will get an error.

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

-- Inserting values to the Colleges table
INSERT INTO Colleges(college_id, college_code, college_name)
VALUES (1, "ARD12", "Star Public School"), (2, "ARD12", "Galaxy School");
```

Run Code »

Here, we are trying to insert **ARD12** in the `college_code` column in two different rows. Hence, the INSERT INTO command results in an error.

CREATE UNIQUE INDEX for Unique Values

If we want to create indexes for unique values in a column, we use the `CREATE UNIQUE INDEX` constraint. For example,

```
-- create unique index
CREATE UNIQUE INDEX college_index
ON Colleges(college_code);
```

Run Code »

Here, the SQL command creates a unique index named `college_index` on the `Colleges` table using the `college_code` column.

> **Note:** Although the index is created for only unique values, the original data in the table remains unaltered.

Recommended Readings

- SQL CREATE INDEX
- SQL Constraints
- SQL NOT NULL Constraint
- SQL PRIMARY KEY

Previous Tutorial: SQL Not Null Constraint

Next Tutorial: SQL Primary Key

Did you find this article helpful?

Related Tutorials

Programming — SQL CREATE INDEX

Programming — SQL PRIMARY KEY

Programming — SQL Constraints

Programming — SQL NOT NULL Constraint