Creating and Altering Tables in MySQL

Build and manage your database efficiently!

Creating a Table in MySQL

Tables are the backbone of any relational database. Here's how you can create one in MySQL:

```
CREATE TABLE table_name ( column1 datatype
constraints, column2 datatype constraints, ... );
```

Example

```
CREATE TABLE employees (
employee_id INT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
    hire_date DATE,
 salary DECIMAL(10, 2));
```

The employees table includes an ID (primary key), first and last names, hire date, and salary fields. This is the foundation of most databases!

Adding Constraints to Your Table

Constraints help maintain data integrity by enforcing rules at the column level.

```
CREATE TABLE table_name (
column1 datatype NOT NULL UNIQUE,
column2 datatype CHECK (condition), ...);
```

Example

```
CREATE TABLE departments (
department_id INT PRIMARY KEY,
department_name VARCHAR(50) NOT
NULL UNIQUE,
manager_id INT CHECK (manager_id > 0));
```

Constraints such as NOT NULL, UNIQUE, and CHECK ensure data integrity by enforcing rules. In the departments table, department names must be unique, and manager IDs must be greater than 0.

Altering a Table - Adding Columns

As your database grows, you may need to add more fields to your tables. Use the ALTER TABLE command for that!

```
ALTER TABLE table_name
ADD column_name datatype;
```

Example

```
ALTER TABLE employees
ADD department_id INT;
```

This command adds a new department_id column to the employees table, allowing us to link employees to departments. This helps keep data relational and organized.

Altering a Table - Modifying Columns

Need to change the data type or size of a column? You can modify it using ALTER TABLE:

```
ALTER TABLE table_name
 MODIFY column_name new_datatype;
```

Example

```
ALTER TABLE employees
MODIFY salary DECIMAL(15, 2);
```

We modified the salary column to increase its precision for larger salary values. This command is essential when data requirements change over time.

Renaming Columns or Tables

Need to rename a column or table to keep things more organized?

Renaming Column: ALTER TABLE table_name RENAME COLUMN old_name TO new_name;
Renaming Table: RENAME TABLE old_table_name TO new_table_name;

Example

```
ALTER TABLE employees
RENAME COLUMN first_name TO fname;
```

This command renames the first_name column to fname. Renaming helps improve clarity or reflect updated naming conventions as your project evolves.

Dropping Columns or Tables

Need to remove outdated columns or tables? Use the DROP command:

```
ALTER TABLE table_name
DROP COLUMN column_name


DROP TABLE table_name;
```

Example

```
ALTER TABLE employees
DROP COLUMN hire_date;
DROP TABLE employees;
```

Dropping a column or table permanently removes data, so use it cautiously. This is handy for getting rid of outdated fields or tables when data is no longer needed.

Practice Table Creation & Alteration

- Create a products table with product_id, name, category, and price.
- Add a stock_quantity column to the products table.
- Modify the price column to increase its precision.
- Drop the category column if no longer needed.

```
-- Create the products table
CREATE TABLE products (
product_id INT PRIMARY KEY, name VARCHAR(50),
category VARCHAR(50), price DECIMAL(10, 2));
-- Add a new column
ALTER TABLE products
ADD stock_quantity INT;
-- Modify the price column
ALTER TABLE products
MODIFY price DECIMAL(15, 2);
-- Drop the category column
ALTER TABLE products
DROP COLUMN category;
```

Master SQL Table Operations!

Now that you know how to create and alter tables, it's time to practice! Try out these commands on your own database and see how they help you manage and organize your data better.