MySQL Database

MySQL CREATE DATABASE Statement

The MySQL CREATE DATABASE Statement

The CREATE DATABASE statement is used to create a new SQL database.

Syntax

CREATE DATABASE *databasename*;

CREATE DATABASE Example

The following SQL statement creates a database called "testDB":

Example

CREATE DATABASE testDB;

MySQL DROP DATABASE Statement

The MySQL DROP DATABASE Statement

The DROP DATABASE statement is used to drop an existing SQL database.

Syntax

DROP DATABASE *databasename*;

Note: Be careful before dropping a database. Deleting a database will result in loss of complete information stored in the database!

DROP DATABASE Example

The following SQL statement drops the existing database "testDB":

Example

DROP DATABASE testDB;

MySQL CREATE TABLE Statement

The MySQL CREATE TABLE Statement

The CREATE TABLE statement is used to create a new table in a database.

Syntax

```
CREATE TABLE table_name (
   column1 datatype,
   column2 datatype,
   column3 datatype,
  ....
);
```

The column parameters specify the names of the columns of the table.

The datatype parameter specifies the type of data the column can hold (e.g. varchar, integer, date, etc.).

Tip: For an overview of the available data types, go to our complete Data Types Reference.

MySQL CREATE TABLE Example

The following example creates a table called "Persons" that contains five columns: PersonID, LastName, FirstName, Address, and City:

Example

```
CREATE TABLE Persons (
   PersonID int,
   LastName varchar(255),
   FirstName varchar(255),
   Address varchar(255),
   City varchar(255)
);
```

MySQL DROP TABLE Statement

The MySQL DROP TABLE Statement

The DROP TABLE statement is used to drop an existing table in a database.

Syntax

DROP TABLE *table_name*;

Note: Be careful before dropping a table. Deleting a table will result in loss of complete information stored in the table!

MySQL DROP TABLE Example

The following SQL statement drops the existing table "Shippers":

Example

DROP TABLE Shippers;

MySQL ALTER TABLE Statement

MySQL ALTER TABLE Statement

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

ALTER TABLE *table_name*
ADD *column_name datatype*;

The following SQL adds an "Email" column to the "Customers" table:

Example

ALTER TABLE Customers
ADD Email varchar(255);

ALTER TABLE - DROP COLUMN

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

ALTER TABLE *table_name*
DROP COLUMN *column_name*;

The following SQL deletes the "Email" column from the "Customers" table:

Example

```
ALTER TABLE Customers
DROP COLUMN Email;
```

ALTER TABLE - MODIFY COLUMN

To change the data type of a column in a table, use the following syntax:

ALTER TABLE *table_name*
MODIFY COLUMN *column_name datatype*;

MySQL NOT NULL Constraint

By default, a column can hold NULL values.

The NOT NULL constraint enforces a column to NOT accept NULL values.

This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

NOT NULL on CREATE TABLE

The following SQL ensures that the "ID", "LastName", and "FirstName" columns will NOT accept NULL values when the "Persons" table is created:

Example

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255) NOT NULL,
   Age int
);
```

NOT NULL on ALTER TABLE

To create a NOT NULL constraint on the "Age" column when the "Persons" table is already created, use the following SQL:

Example

```
ALTER TABLE Persons
MODIFY Age int NOT NULL;
```

MySQL UNIQUE Constraint

The UNIQUE constraint ensures that all values in a column are different.

Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.

A PRIMARY KEY constraint automatically has a UNIQUE constraint.

However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

UNIQUE Constraint on CREATE TABLE

The following SQL creates a UNIQUE constraint on the "ID" column when the "Persons" table is created:

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   UNIQUE (ID)
);
```

To name a UNIQUE constraint, and to define a UNIQUE constraint on multiple columns, use the following SQL syntax:

```
CREATE TABLE Persons (
   ID int NOT NULL,
   LastName varchar(255) NOT NULL,
   FirstName varchar(255),
   Age int,
   CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```
