**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; |

**Tip:** Make sure you have admin privilege before creating any database. Once a database is created, you can check it in the list of databases with the following SQL command: SHOW DATABASES;

**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; |
| **Tip:** Make sure you have admin privilege before dropping any database. Once a database is dropped, you can check it in the list of databases with the following SQL command: SHOW DATABASES; |
| **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*, .... );

**Create Table Using Another Table**

A copy of an existing table can also be created using CREATE TABLE.

The new table gets the same column definitions. All columns or specific columns can be selected.

If you create a new table using an existing table, the new table will be filled with the existing values from the old table.

# Syntax

CREATE TABLE *new\_table\_name* AS SELECT *column1, column2,...*

FROM *existing\_table\_name*

WHERE ....;

The following SQL creates a new table called "TestTables" (which is a copy of the "Customers" table):

|  |
| --- |
| **Example** |
| CREATE TABLE TestTable AS |

SELECT customername, contactname FROM customers;

**The MySQL DROP TABLE Statement**

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

# Syntax

DROP TABLE *table\_name*;

**MySQL DROP TABLE Example**

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

|  |
| --- |
| **Example** |
| DROP TABLE Shippers; |

**MySQL TRUNCATE TABLE**

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

# Syntax

TRUNCATE TABLE *table\_name*;

**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**

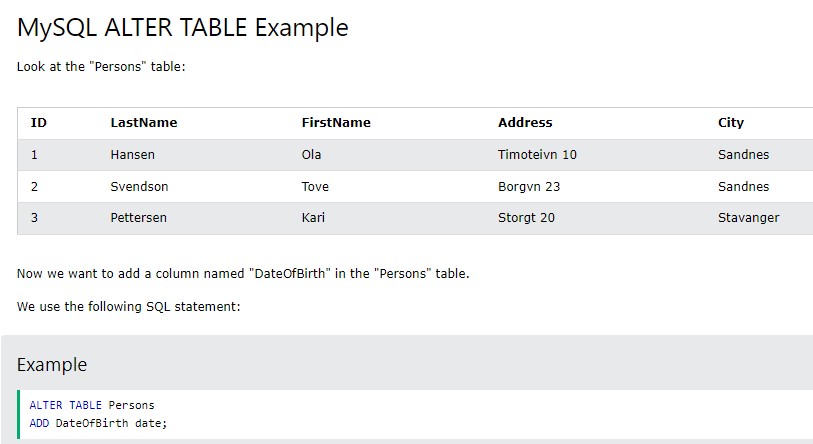
To change the data type of a column in a table, use the following syntax:

ALTER TABLE *table\_name*

MODIFY COLUMN *column\_name datatype*;

Notice that the new column, "DateOfBirth", is of type date and is going to hold a date. The data type specifies what type of data the column can hold. For a complete reference of all the data types available in MySQL, go to our complete Data Types reference.

The "Persons" table will now look like this:



Notice that the new column, "DateOfBirth", is of type date and is going to hold a date. The data type specifies what type of data the column can hold. For a complete reference of all the data types available in MySQL, go to our complete Data Types reference.

The "Persons" table will now look like this:



**Change Data Type Example**

Now we want to change the data type of the column named "DateOfBirth" in the "Persons" table.

We use the following SQL statement:

|  |
| --- |
| **Example** |
| ALTER TABLE Persons |

MODIFY COLUMN DateOfBirth year;

Notice that the "DateOfBirth" column is now of type year and is going to hold a year in a two- or four-digit format.

**DROP COLUMN Example**

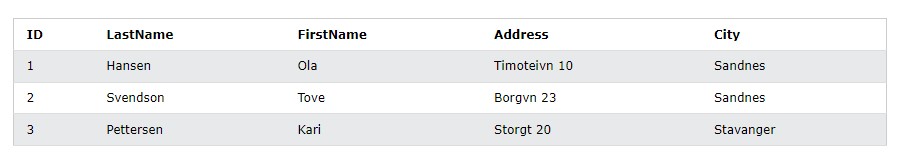
Next, we want to delete the column named "DateOfBirth" in the "Persons" table.

We use the following SQL statement:

|  |
| --- |
| **Example** |
| ALTER TABLE Persons |

DROP COLUMN DateOfBirth;

The "Persons" table will now look like this:



MySQL Constraints

SQL constraints are used to specify rules for data in a table.

**Create Constraints**

Constraints can be specified when the table is created with the CREATE

TABLE statement, or after the table is created with the ALTER TABLE statement.

# Syntax

CREATE TABLE *table\_name* (  *column1 datatype* *constraint*,  *column2 datatype* *constraint*,  *column3 datatype* *constraint*, .... );

**MySQL Constraints**

SQL constraints are used to specify rules for the data in a table.

Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the table. If there is any violation between the constraint and the data action, the action is aborted.

Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.

The following constraints are commonly used in SQL:

* **NOT NU**LL - Ensures that a column cannot have a NULL value
* **UNIQUE** - Ensures that all values in a column are different
* **PRIMAR**Y KEY - A combination of a NOT NULL and UNIQUE. Uniquely **identif**ies each row in a table
* **FOREIG**N KEY - Prevents actions that would destroy links between tables
* **CHECK** - Ensures that the values in a column satisfies a specific **conditi**on
* **DEFAUL**T - Sets a default value for a column if no value is specified
* **CREATE** INDEX - Used to create and retrieve data from the database very **quickl**y

**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) );

**UNIQUE Constraint on ALTER TABLE**

To create a UNIQUE constraint on the "ID" column when the table is already created, use the following SQL:

ALTER TABLE Persons

ADD UNIQUE (ID);

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

ALTER TABLE Persons

ADD CONSTRAINT UC\_Person UNIQUE (ID,LastName);

**DROP a UNIQUE Constraint**

To drop a UNIQUE constraint, use the following SQL:

ALTER TABLE Persons

DROP INDEX UC\_Person;

**MySQL PRIMARY KEY Constraint**

The PRIMARY KEY constraint uniquely identifies each record in a table.

Primary keys must contain UNIQUE values, and cannot contain NULL values.

A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

**PRIMARY KEY on CREATE TABLE**

The following SQL creates a PRIMARY KEY 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,

PRIMARY KEY (ID) );

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY 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 PK\_Person PRIMARY KEY (ID,LastName) );

**Note:** In the example above there is only ONE PRIMARY KEY (PK\_Person). However, the VALUE of the primary key is made up of TWO COLUMNS (ID + LastName).

**PRIMARY KEY on ALTER TABLE**

To create a PRIMARY KEY constraint on the "ID" column when the table is already created, use the following SQL:

ALTER TABLE Persons

ADD PRIMARY KEY (ID);

To allow naming of a PRIMARY KEY constraint, and for defining a PRIMARY KEY constraint on multiple columns, use the following SQL syntax:

ALTER TABLE Persons

ADD CONSTRAINT PK\_Person PRIMARY KEY (ID,LastName);

**Note:** If you use ALTER TABLE to add a primary key, the primary key column(s) must have been declared to not contain NULL values (when the table was first created).

**DROP a PRIMARY KEY Constraint**

To drop a PRIMARY KEY constraint, use the following SQL:

ALTER TABLE Persons DROP PRIMARY KEY;

**MySQL CHECK Constraint**

The CHECK constraint is used to limit the value range that can be placed in a column.

If you define a CHECK constraint on a column it will allow only certain values for this column.

If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

**CHECK on CREATE TABLE**

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created. The CHECK constraint ensures that the age of a person must be 18, or older:

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

CHECK (Age>=18)

);

To allow naming of a CHECK constraint, and for defining a CHECK 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,

City varchar(255),

CONSTRAINT CHK\_Person CHECK (Age>=18 AND City='Sandnes') );

**CHECK on ALTER TABLE**

To create a CHECK constraint on the "Age" column when the table is already created, use the following SQL:

ALTER TABLE Persons

ADD CHECK (Age>=18);

To allow naming of a CHECK constraint, and for defining a CHECK constraint on multiple columns, use the following SQL syntax:

ALTER TABLE Persons

ADD CONSTRAINT CHK\_PersonAge CHECK (Age>=18 AND City='Sandnes');

**DROP a CHECK Constraint**

To drop a CHECK constraint, use the following SQL:

ALTER TABLE Persons

DROP CHECK CHK\_PersonAge;

**MySQL DEFAULT Constraint**

The DEFAULT constraint is used to set a default value for a column.

The default value will be added to all new records, if no other value is specified.

**DEFAULT on CREATE TABLE**

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

CREATE TABLE Persons (

ID int NOT NULL,

LastName varchar(255) NOT NULL,

FirstName varchar(255),

Age int,

City varchar(255) DEFAULT 'Sandnes'

);

The DEFAULT constraint can also be used to insert system values, by using functions like CURRENT\_DATE():

CREATE TABLE Orders (

ID int NOT NULL,

OrderNumber int NOT NULL,

OrderDate date DEFAULT CURRENT\_DATE() );

**DEFAULT on ALTER TABLE**

To create a DEFAULT constraint on the "City" column when the table is already created, use the following SQL:

ALTER TABLE Persons

ALTER City SET DEFAULT 'Sandnes';

**DROP a DEFAULT Constraint**

To drop a DEFAULT constraint, use the following SQL:

ALTER TABLE Persons

ALTER City DROP DEFAULT;