**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 NULL](https://www.w3schools.com/MySQL/mysql_notnull.asp) - Ensures that a column cannot have a NULL value
* [UNIQUE](https://www.w3schools.com/MySQL/mysql_unique.asp) - Ensures that all values in a column are different
* [PRIMARY KEY](https://www.w3schools.com/MySQL/mysql_primarykey.asp) - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
* [FOREIGN KEY](https://www.w3schools.com/MySQL/mysql_foreignkey.asp) - Prevents actions that would destroy links between tables
* [CHECK](https://www.w3schools.com/MySQL/mysql_check.asp) - Ensures that the values in a column satisfies a specific condition
* [DEFAULT](https://www.w3schools.com/MySQL/mysql_default.asp) - Sets a default value for a column if no value is specified
* [CREATE INDEX](https://www.w3schools.com/MySQL/mysql_create_index.asp) - Used to create and retrieve data from the database very quickly

# MySQL NOT NULL Constraint

## 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  
);

CREATE table users (id int AUTO\_INCREMENT PRIMARY KEY, fname varchar(20) not null, lname varchar(20) not null, city varchar(20))

INSERT into users (fname, lname, city) values ('bhavin', 'thakar', 'rajkot')

INSERT into users (fname, lname, city) values ('bhavya', 'dava', null);

## 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;

ALTER TABLE users MODIFY COLUMN city varchar(20) not null

INSERT into users (fname, lname, city) values ('bhavya', 'dava', null);

#1048 - Column 'city' cannot be null

# MySQL UNIQUE Constraint

## 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.

DROP TABLE users

## 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)  
);

CREATE TABLE users (id int AUTO\_INCREMENT PRIMARY key, fname varchar(20) not null, lname varchar(20) not null, city varchar(20) not null, email varchar(64) UNIQUE)

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', 'het@gmail.com')

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', 'het@gmail.com')

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', null)

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', null);

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', null);

## 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 TABLE users

CREATE TABLE users (id int AUTO\_INCREMENT PRIMARY key, fname varchar(20) not null, lname varchar(20) not null, city varchar(20) not null, email varchar(64) not null)

ALTER TABLE users MODIFY COLUMN email varchar(64) not null UNIQUE

alter TABLE users add CONSTRAINT unq\_email UNIQUE(email)

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', 'het@gmail.com')

INSERT into users (fname, lname, city, email) VALUES ('het', 'manani', 'rajkot', 'het@gmail.com')

#1062 - Duplicate entry 'het@gmail.com' for key 'unq\_email'

## DROP a UNIQUE Constraint

To drop a UNIQUE constraint, use the following SQL:

ALTER TABLE Persons  
DROP INDEX UC\_Person;

ALTER TABLE users drop CONSTRAINT unq\_email

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)  
);

create table listofvillage (villageid int AUTO\_INCREMENT PRIMARY key, villagename varchar(20) not null, cityname varchar(20) not null, taluka varchar(20) not null, district varchar(20) not null, state varchar(20) not null, CONSTRAINT unq\_villagename UNIQUE(villagename, cityname, taluka))

INSERT into listofvillage (villagename, cityname, taluka, district, state) VALUES('Navagam', 'Rajkot', 'Rajkot', 'Rajkot', 'Gujarat')

INSERT into listofvillage (villagename, cityname, taluka, district, state) VALUES('Navagam', 'Rajkot', 'Rajkot', 'Rajkot', 'Gujarat');

INSERT into listofvillage (villagename, cityname, taluka, district, state) VALUES('Navagam', 'Gondal', 'Rajkot', 'Rajkot', 'Gujarat');

# MySQL PRIMARY KEY Constraint

## 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).

CREATE TABLE students (id int PRIMARY key, fname varchar(20) not null, lname varchar(20) not null, email varchar(64) not null)

INSERT into students (id, fname, lname, email) values (1, 'keyur', 'ramani', 'keyur@rajkot.com')

INSERT into students (id, fname, lname, email) values (1, 'keyur', 'ramani', 'keyur@rajkot.com')

#1062 - Duplicate entry '1' for key 'PRIMARY'

## DROP a PRIMARY KEY Constraint

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

ALTER TABLE Persons  
DROP PRIMARY KEY;

alter TABLE students drop PRIMARY key

## 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);

alter TABLE students add PRIMARY key (id)

TRUNCATE TABLE students

alter table students add CONSTRAINT pri\_key\_id PRIMARY key (id)

INSERT into students (id, fname, lname, email) values (1, 'keyur', 'ramani', 'keyur@rajkot.com');

INSERT into students (id, fname, lname, email) values (1, 'keyur', 'ramani', 'keyur@rajkot.com')

#1062 - Duplicate entry '1' for key 'PRIMARY'

# MySQL FOREIGN KEY Constraint

## MySQL FOREIGN KEY Constraint

The FOREIGN KEY constraint is used to prevent actions that would destroy links between tables.

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the [PRIMARY KEY](https://www.w3schools.com/MySQL/mysql_primarykey.asp) in another table.

The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

## FOREIGN KEY on CREATE TABLE

The following SQL creates a FOREIGN KEY on the "PersonID" column when the "Orders" table is created:

CREATE TABLE Orders (  
    OrderID int NOT NULL,  
    OrderNumber int NOT NULL,  
    PersonID int,  
    PRIMARY KEY (OrderID),  
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)  
);

CREATE TABLE students (id int AUTO\_INCREMENT PRIMARY key, fname varchar(20) not null, lname varchar(20) not null, city varchar(20) not null)

INSERT into students (fname, lname, city) VALUES ('het', 'Manani', 'rajkot'),('bhavin', 'thakar', 'rajkot'),('bhavya', 'dava', 'rajkot'),('rajiv', 'shekh', 'rajkot'),('yash', 'singla', 'rajkot'),('yash', 'chavda', 'rajkot'),('keyur', 'ramani', 'rajkot'),('prince', 'gajirapa', 'rajkot'),('kalpes', 'chauahn', 'rajkot'),('het', 'Manani', 'rajkot');

CREATE TABLE attendance (attendance\_id int AUTO\_INCREMENT PRIMARY key, students\_id int, absents int, presents int, FOREIGN key (students\_id) REFERENCES students(id))

drop TABLE attendance

CREATE TABLE attendance (attendance\_id int AUTO\_INCREMENT PRIMARY key, students\_id int, absents int, presents int, CONSTRAINT fk\_students\_attendance\_id FOREIGN key (students\_id) REFERENCES students(id))

INSERT into attendance (students\_id, absents, presents) VALUES (1, 111, 112)

INSERT into attendance (students\_id, absents, presents) VALUES (111, 111, 112);

#1452 - Cannot add or update a child row: a foreign key constraint fails (`224sample`.`attendance`, CONSTRAINT `fk\_students\_attendance\_id` FOREIGN KEY (`students\_id`) REFERENCES `students` (`id`))

ALTER TABLE attendance drop CONSTRAINT fk\_students\_attendance\_id

ALTER TABLE attendance add CONSTRAINT fk\_students\_attendance\_id FOREIGN key (students\_id) REFERENCES students (id);

# MySQL CHECK Constraint

## 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)  
);

create table users (id int AUTO\_INCREMENT PRIMARY key, fname varchar(20), lname varchar(20), city varchar(20), email varchar(64), age int check (age >= 18))

INSERT into users (fname, lname, city, email, age) VALUES ('Bhavya', 'Dava', 'Rajkot', 'bhavya@gmail.com', 19)

INSERT into users (fname, lname, city, email, age) VALUES ('Bhavya', 'Dava', 'Rajkot', 'bhavya@gmail.com', 17);

alter TABLE users add CONSTRAINT check\_city CHECK (city = 'Rajkot' or city = 'Morbi' or city = 'surat')

INSERT into users (fname, lname, city, email, age) VALUES ('Bhavya', 'Dava', 'Rajkot', 'bhavya@gmail.com', 19)

INSERT into users (fname, lname, city, email, age) VALUES ('Bhavya', 'Dava', 'Bhuj', 'bhavya@gmail.com', 19);

## 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);

## DROP a CHECK Constraint

To drop a CHECK constraint, use the following SQL:

ALTER TABLE Persons  
DROP CHECK CHK\_PersonAge;

ALTER TABLE users drop CONSTRAINT check\_city

# MySQL DEFAULT Constraint

## 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'  
);

drop TABLE users

CREATE TABLE users (id int AUTO\_INCREMENT PRIMARY KEY, fname varchar(20), lname varchar(20), city varchar(20) DEFAULT 'Rajkot', create\_at timestamp DEFAULT CURRENT\_TIMESTAMP, updated\_at timestamp DEFAULT CURRENT\_TIMESTAMP on UPDATE CURRENT\_TIMESTAMP)

INSERT into users (fname, lname) VALUES ('keyur', 'ramani')

INSERT into users (fname, lname, city) VALUES ('keyur', 'ramani', 'Baroda')

## 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';

ALTER TABLE users ALTER lname set DEFAULT 'Not Required'

INSERT into users (fname) VALUES ('Het');

## DROP a DEFAULT Constraint

To drop a DEFAULT constraint, use the following SQL:

ALTER TABLE Persons  
ALTER City DROP DEFAULT;

ALTER TABLE users ALTER city drop DEFAULT

# MySQL CREATE INDEX Statement

## MySQL CREATE INDEX Statement

The CREATE INDEX statement is used to create indexes in tables.

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

**Note:** Updating a table with indexes takes more time than updating a table without (because the indexes also need an update). So, only create indexes on columns that will be frequently searched against.

### CREATE INDEX Syntax

Creates an index on a table. Duplicate values are allowed:

CREATE INDEX index\_name  
ON table\_name (column1, column2, ...);

### CREATE UNIQUE INDEX Syntax

Creates a unique index on a table. Duplicate values are not allowed:

CREATE UNIQUE INDEX index\_name  
ON table\_name (column1, column2, ...);

CREATE index fnameindex on students (fname)

CREATE UNIQUE index fname\_index on students (fname)

## DROP INDEX Statement

The DROP INDEX statement is used to delete an index in a table.

ALTER TABLE table\_nameDROP INDEX index\_name;

DROP INDEX fnameindex on students

# MySQL AUTO INCREMENT Field

## What is an AUTO INCREMENT Field?

Auto-increment allows a unique number to be generated automatically when a new record is inserted into a table.

Often this is the primary key field that we would like to be created automatically every time a new record is inserted.

## MySQL AUTO\_INCREMENT Keyword

MySQL uses the AUTO\_INCREMENT keyword to perform an auto-increment feature.

By default, the starting value for AUTO\_INCREMENT is 1, and it will increment by 1 for each new record.

drop TABLE users

CREATE TABLE users (id int AUTO\_INCREMENT PRIMARY key, fname varchar(20), lname varchar(20))

INSERT into users (fname, lname) VALUES ('het', 'manani')

INSERT into users (fname, lname) VALUES ('bhavin', 'thakar');

alter TABLE users AUTO\_INCREMENT = 101

INSERT into users (fname, lname) VALUES ('bhavya', 'dava');

# MySQL Working With Dates

## MySQL Dates

The most difficult part when working with dates is to be sure that the format of the date you are trying to insert, matches the format of the date column in the database.

As long as your data contains only the date portion, your queries will work as expected. However, if a time portion is involved, it gets more complicated.

## MySQL Date Data Types

MySQL comes with the following data types for storing a date or a date/time value in the database:

* DATE - format YYYY-MM-DD
* DATETIME - format: YYYY-MM-DD HH:MI:SS
* TIMESTAMP - format: YYYY-MM-DD HH:MI:SS
* YEAR - format YYYY or YY

**Note:** The date data type are set for a column when you create a new table in your database!

SELECT \* from students WHERE dob > '2005-01-01'

SELECT \* from students WHERE dob = '2005-10-07';

SELECT \* from students WHERE created\_at = '2024-10-07'

SELECT \* from students WHERE created\_at like '2024-10-07%';

SELECT \* from students WHERE date(created\_at) = '2024-10-07';

**Tip:** To keep your queries simple and easy to maintain, do not use time-components in your dates, unless you have to!

# MySQL Views

## MySQL CREATE VIEW Statement

In SQL, a view is a virtual table based on the result-set of an SQL statement.

A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

A view is created with the CREATE VIEW statement.

### CREATE VIEW Syntax

CREATE VIEW view\_name AS  
SELECT column1, column2, ...  
FROM table\_name  
WHERE condition;

**Note:** A view always shows up-to-date data! The database engine recreates the view, every time a user queries it.

CREATE view v1 as SELECT \* from students

SELECT \* FROM `v1`

SELECT \* FROM `v1` WHERE id = 5

CREATE or REPLACE view v1 as SELECT id, fname, lname FROM students

## MySQL Dropping a View

A view is deleted with the DROP VIEW statement.

### DROP VIEW Syntax

DROP VIEW view\_name;

drop view v1