**1) To create a new database in MySQL,**

**you use the CREATE DATABASE statement with the following syntax:**

**CREATE DATABASE database\_name;**

**CREATE DATABASE testdb;**

**3) Finally, to access the newly created database, you use the USE database command as follows:**

**USE testdb;**

**4) DROP DATABASE statement to delete an existing database in the server.**

**The DROP DATABASE statement drops all tables in the database and deletes the database permanently.**

**Therefore, you should be very careful when using this statement.**

**DROP DATABASE database\_name;**

**5)After creating table to view table\_structure**

**sp\_help tablename;**

**sp\_help trainer;**

## 6) The SQL 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*,  
   ....  
);**

**Or**

**create table cloneEmployee**

**as select \* from employees;**

# SQL Constraints

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

## SQL 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*,  
    ....  
);**

## SQL 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/sql/sql_notnull.asp)**- Ensures that a column cannot have a NULL value**
* [**UNIQUE**](https://www.w3schools.com/sql/sql_unique.asp)**- Ensures that all values in a column are different**
* [**PRIMARY KEY**](https://www.w3schools.com/sql/sql_primarykey.asp)**- A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table**
* [**FOREIGN KEY**](https://www.w3schools.com/sql/sql_foreignkey.asp)**- Prevents actions that would destroy links between tables**
* [**DEFAULT**](https://www.w3schools.com/sql/sql_default.asp)**- Sets a default value for a column if no value is specified**
* **CHECK**

## create table trainer

## (tid int primary key ,

## tname varchar(25) not null,

## mobile bigint unique ,

## email varchar(25) ,

## qualification varchar(25) default "BE"

## );

## 7) SQL 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/sql/sql_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.**

**[CONSTRAINT constraint\_name]**

**FOREIGN KEY (column\_name, ...)**

**REFERENCES parent\_table(colunm\_name,...)**

**[ON DELETE reference\_option]**

**[ON UPDATE reference\_option]**

**SQLServer has reference options: CASCADE, SET NULL,RESTRICT.**

**If you don’t specify the ON DELETE and ON UPDATE clause,**

**the default action is RESTRICT.**

* **CASCADE: if a row from the parent table is deleted or updated,**

**the values of the matching rows in the child table automatically**

**deleted or updated.**

* **SET NULL: if a row from the parent table is deleted or updated,**

**the values of the foreign key column (or columns) in the child table**

**are set to NULL.**

* **RESTRICT: if a row from the parent table has a matching row in the**

**child table, MySQL rejects deleting or updating rows in the parent table.**

**CREATE TABLE Trainer (**

**trainerid INT PRIMARY KEY,**

**trainername VARCHAR(45) NOT NULL,**

**mobile BIGINT unique );**

**CREATE TABLE Batch (**

**batchid INT identity(101,1) PRIMARY KEY,**

**batchname VARCHAR(45) NOT NULL,**

**trainerid int,**

**CONSTRAINT FK\_trainerid**

**FOREIGN KEY(trainerid)**

**REFERENCES trainer(trainerid)**

**ON UPDATE RESTRICT ON DELETE set null**

**);**

**Alter table batch**

**Drop constraint fk\_trainerid;**

**Delete from trainer where trainerid=1**

**CREATE TABLE Student (**

**sid INT identity(1,1),**

**sname VARCHAR(45) NOT NULL,**

**dob Date,**

**mobile INT unique,**

**batchid int,**

**Primary key(sid),**

**FOREIGN KEY (batchid)**

**REFERENCES batch(batchid)**

**ON UPDATE RESTRICT ON DELETE CASCADE**

**);**

**7) INSERT INTO student(sname,dob,mobile)**

**VALUES('Mohit','2001-01-09',9890023456);**

**8) SELECT \* FROM Student;**

**//Composite key**

**CREATE TABLE Test (**

**testid INT,**

**sid INT,**

**marks *int NOT NULL*,**

**PRIMARY KEY (testid,sid),**

**FOREIGN KEY (sid)**

**REFERENCES Student(sid)**

**ON UPDATE RESTRICT ON DELETE CASCADE**

**);**

**create table test**

**(tid int ,**

**sid int,**

**marks int,**

**primary key(tid,sid),**

**foreign key(sid) references student(sid),**

**foreign key(tid) references Test(tid)**

**);**

## 9) SQL 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 Student**

**ADD email VARCHAR(45);**

**ALTER TABLE Student**

**Alter column mobile bigint;**

**ALTER TABLE table\_name**

**DROP COLUMN column\_name;**

**ALTER TABLE table\_name**

**DROP constraint constraint\_name;**

**ALTER TABLE student**

**ADD PRIMARY KEY(sid);**

**ALTER TABLE Student**

**ADD FOREIGN KEY (batchid) REFERENCES Batch(batchID);**

**ALTER TABLE Student**

**ADD CONSTRAINT FK\_batchid**

**FOREIGN KEY (batchid) REFERENCES Batch(batchID);**

**Rename column name in MS SQL Server**

The process of renaming column name is MS SQL Server is different when compared to the other databases. In MS SQL Server, you have to use the stored procedure called **sp\_rename.**

**Syntax**

|  |  |
| --- | --- |
| 1 | sp\_rename 'TableName.OldColumnName', 'New ColumnName', 'COLUMN'; |

**Example:**

Write a query to rename the column name “BID” to “BooksID”.

|  |  |
| --- | --- |
| 1 | sp\_rename 'Books.BID', 'BooksID', 'COLUMN'; |

The resulting output will be the same as that for the above queries. Now, that you have understood how to rename a column name in various databases, let us see how you can rename a table name.

**To change one or more tables, we use the statement as follows:**

**sp\_rename 'old\_table\_name', 'new\_table\_name'**

**Introduction to SQL primary key**

**A primary key is a column or a set of columns that uniquely identifies**

**each row in the table.**

**The primary key follows these rules:**

**A primary key must contain unique values. If the primary key consists of**

**multiple columns, the combination of values in these columns must be unique**

**A primary key column cannot have NULL values. Any attempt to insert or**

**update NULL to primary key columns will result in an error.**

**Note that SQL implicitly adds a NOT NULL constraint to primary key**

**columns.**

**A table can have one an only one primary key.**

**mysql primary key**

**Because SQL works faster with integers, the data type of the primary key**

**column should be the integer e.g., INT, BIGINT.**

**And you should ensure sure that value ranges of the integer type for the**

**primary key are sufficient for storing all possible rows that the table**

**may have.**

**A primary key column often has the identity(start,increment) attribute that**

**automatically generates a sequential integer whenever you insert a new row**

**into the table.**

**When you define a primary key for a table,**

**SQL automatically creates an index called PRIMARY.**

**The SQL UPDATE Statement:**

**The UPDATE statement is used to modify the existing records in a table.**

**UPDATE Syntax**

**UPDATE table\_name**

**SET column1 = value1, column2 = value2, ...**

**WHERE condition;**

**Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!**

**Example:**

**UPDATE Student  
SET studname = 'Mohit', City= 'Pune'  
WHERE sid = 1;**

**The SQL DELETE Statement**

**The DELETE statement is used to delete existing records in a table.**

**DELETE Syntax**

**DELETE FROM *table\_name*WHERE *condition*;**

**Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!**

**Example:**

**DELETE FROM Student WHERE studname='Mohit';**

**SELECT \* FROM testdb.employees;**

**Select \* from student where city in ('pune','nashik');**

**select \* from employees where salary>10000;**

**select \* from employees where salary>=10000 and salary<=20000;**

**select \* from employees where salary between 10000 and 20000;**

**select \* from employees where phone\_number is not null;**

**SELECT \* FROM testdb.employees where first\_name like '\_a\_s\_';**

**select \* from employees where first\_name like ‘A%';**

**select \* from employees order by department\_id desc ,first\_name desc;**

**drop table student;**

**drop database thinkQ;**

**select \* from student;**

**select \* from student where qualification=’Mech’ or qualification=’CS’**

**select \* from student where qualification in (‘Mech’,’CS’,’Elect’);**

**select \* from product where price between 1000 and 2000;**

**select \* from student where mobile is null;**

**select \* from student where mobile is not null;**

select distinct city from student;

insert into student(sid,sname,mobile,city,courseid,experiene,percentage) values(19,'Palash',98900989,'Pune',1,10,90);

select top 1 \* from student order by percentage asc;

select city,count(\*) as countstud from student where percentage>=60 group by city order by countstud;

select city,count(\*) as countstud from student where percentage>=60 and experiene>=3 and courseid=1 group by city order by countstud;

select cid from course where coursename='Csharp';

select \* from student where courseid in(select cid from course where coursename='Csharp');

select \* from student where batchid in (select batchid from batch where trainerid=(select tid from trainer where trainername='Deepa'))

delete from student where batchid=(select batchid from batch where batchname='musk');

update student

set percentage=percentage+5

where batchid=(select batchid from batch where batchname='Steve')

select \* from student, batch;

select \* from student cross join batch;

select \* from student inner join batch on student.batchid=batch.batchid;

select \* from emp e right join dept d on e.deptid=d.deptid;

select \* from emp e left join dept d on e.deptid=d.deptid;

select \* from dept d right join emp e on e.deptid=d.deptid;

select \* from dept d full outer join emp e on e.deptid=d.deptid;

select e.mgrid,count(\*) from thinkEmp e inner join thinkEmp m on e.mgrid=m.Empid group by e.mgrid ;

select \* from emp e1 where salary>(select avg(salary) from emp e2 where e2.deptid=e1.depid)

select \* from emp order by salary desc offset 2 rows fetch next 2 rows only;

select \* from emp e1 where 3=(select count(\*) from (select distinct salary from emp) as e2 where e2.salary>=e1.salary);

select \* from emp where 3=3;

select \* from emp e1 where 3=(select count(\*) from (select distinct salary from emp) as e2 where e2.salary>=e1.salary);

select sid,sname from student

union

select Empid,Empname from thinkemp;

create index idxpercent on student(percentage);

drop index idxpercent on student;