**mysql -u root -p**

In the syntax, the **-u root indicates** that we will connect to the MySQL server using the root user account and **-p** instructs MySQL to ask for a password.

**SHOW DATABASES;**

**SHOW TABLES;**

When you create a user that already exists, it gives an error. But if you use, **IF NOT EXISTS** clause, the statement gives a warning for each named user that already exists instead of an error message.

1. **CREATE** USER IF NOT EXISTS account\_name IDENTIFIED **BY** 'password';

In the above syntax, the **account\_name** has two parts one is the **username**, and another is the **hostname**, which is separated by **@** symbol. Here, the username is the name of the user, and the hostname is the name of the host from which the user can connect with the database server.

1. username@hostname

hostname is optional. If you have not given the hostname, the user can connect from any host on the server. The user account name without hostname can be written as:

1. username@%

## to show db users

**select** user **from** mysql.user;

## Grant Privileges to the MySQL New User

1. **ALL PRIVILEGES:** It permits all privileges to a new user account.
2. **CREATE:** It enables the user account to create databases and tables.
3. **DROP:** It enables the user account to drop databases and tables.
4. **DELETE:** It enables the user account to delete rows from a specific table.
5. **INSERT:** It enables the user account to insert rows into a specific table.
6. **SELECT:** It enables the user account to read a database.
7. **UPDATE:** It enables the user account to update table rows.

If you want to give all privileges to a newly created user, execute the below command.

**GRANT** ALL **PRIVILEGES** **ON** \* . \* **TO** peter@localhost;

If you want to give specific privileges to a newly created user, execute the command.

**GRANT** **CREATE**, **SELECT**, **INSERT** **ON** \* . \* **TO** peter@localhost;

Sometimes, you want to **flush** all the privileges of a user account for changes occurs immediately, type the following command.

FLUSH **PRIVILEGES**;

If you want to see the existing privileges for the user, execute the following command.

SHOW GRANTS **for** username;

The following syntax is used to delete the user accounts from the database server completely.

**DROP** USER 'account\_name';

Drop multiple user at a time.

**DROP** USER john@localhost, peter@localhost;

#### NOTE: This statement cannot close any open user sessions automatically. In the case when the DROP USER statement executed and the session of this account is active, this statement does not take effect until its session is closed. The user account is dropped only when the session is closed, and that the user's next attempt will not be able to log in again.

If we want to see more information on the user table, execute the command below:

**DESC** mysql.user;

## Show Current User

We can get information of the current user by using the **user() or current\_user()** function, as shown below:

**Select** user(); or **Select** current\_user();

## Show Current Logged User

We can see the currently logged user in the database server by using the below query.

**SELECT** user, host, db, command **FROM** information\_schema.processlist;

# Change MySQL User Password

[MySQL](https://www.javatpoint.com/mysql-tutorial) allows us to change the user account password in three different ways.

1. UPDATE Statement
2. SET PASSWORD Statement
3. ALTER USER Statement

**Change user account password using the UPDATE statement**

1. **UPDATE user SET authentication\_string = PASSWORD('jtp12345') WHERE user = 'peter' AND host = 'localhost';**
2. **FLUSH PRIVILEGES;**

**Change user account password using SET PASSWORD statement**

**SET PASSWORD FOR 'peter'@'localhost' = jtp12345;**

**Change user account password using ALTER USER statement.**

**ALTER USER peter@localhost IDENTIFIED BY 'jtp123';**

We can review the newly created database using the below query that returns the database name, character set, and collation of the database:

**SHOW CREATE DATABASE dbname;**

# MySQL SELECT Database

1. **USE database\_name;**

#### Note: All the database names, table names and table fields name are case sensitive. You must have to use proper names while giving any SQL command.

# MySQL Show/List Databases

**SHOW DATABASES;**

**SHOW SCHEMAS;**

Show Databases command in MySQL also provides an option that allows us to filter the returned database using different pattern matching with LIKE and WHERE clause.

1. SHOW DATABASES LIKE pattern;
2. OR, SHOW DATABASES **WHERE** expression;

3. **SELECT** schema\_name **FROM** information\_schema.schema **WHERE** schema\_name LIKE 's%';

#### NOTE: It is to be noted that if the MySQL server started with the "--skip-show-database" option, we could not use the SHOW DATABASES command unless we have the SHOW DATABASES privilege.

# MySQL DROP Database

**DROP** **DATABASE** [IF EXISTS] database\_name;

# MySQL COPY Database

We need to follow these steps to copy a database to another database:

1. First, use the **CREATE DATABASE** statement to create a new database.
2. Second, store the data to an **SQL file**. We can give any name to this file, but it must end with a **.sql** extension.
3. Third, export all the database objects along with its data to copy using the **mysqldump** tool and then import this file into the new database.

# MySQL CREATE Table

**CREATE** **TABLE** employee\_table( id **int** NOT NULL AUTO\_INCREMENT,

**name** **varchar**(45) NOT NULL, occupation **varchar**(35) NOT NULL,

age **int** NOT NULL, **PRIMARY** **KEY** (id) );

# MySQL ALTER Table

MySQL ALTER statement is used when you want to change the name of your table or any table field. It is also used to add or delete an existing column in a table.

The ALTER statement is always used with "ADD", "DROP" and "MODIFY" commands according to the situation.

## 1) ADD a column in the table

1. **ALTER** **TABLE** table\_name **ADD** new\_column\_name column\_definition

[ **FIRST** | **AFTER** column\_name ];

**2. ALTER** **TABLE** cus\_tbl **ADD** cus\_age **varchar**(40) NOT NULL;

## 2) Add multiple columns in the table

1. **ALTER** **TABLE** cus\_tbl **ADD** cus\_address **varchar**(100) NOT NULL

**AFTER** cus\_surname, **ADD** cus\_salary **int**(100) NOT NULL **AFTER** cus\_age ;

## 3) MODIFY column in the table

1. **ALTER** **TABLE** table\_name **MODIFY** column\_name column\_definition

[ **FIRST** | **AFTER** column\_name ];

**2. ALTER** **TABLE** cus\_tbl **MODIFY** cus\_surname **varchar**(50) NULL;

## 4) DROP column in table

1. **ALTER** **TABLE** table\_name **DROP** **COLUMN** column\_name;
2. **ALTER** **TABLE** cus\_tbl **DROP** **COLUMN** cus\_address;

## 5) RENAME column in table

1. **ALTER** **TABLE** table\_name CHANGE **COLUMN** old\_name new\_name

column\_definition [ **FIRST** | **AFTER** column\_name ]

2. **ALTER** **TABLE** cus\_tbl CHANGE **COLUMN** cus\_surname cus\_title **varchar**(20) NOT NULL;

1. **ALTER** **TABLE** table\_name RENAME **COLUMN** old\_column\_name **TO** new\_column\_name;

## 6) RENAME table

1. **ALTER** **TABLE** table\_name RENAME **TO** new\_table\_name;
2. **ALTER** **TABLE** cus\_tbl RENAME **TO** cus\_table;

# MySQL Show/List Tables

We can also use the **FULL modifier** with the SHOW TABLES query to get the type of table (Base or View) that appears in a second output column.

SHOW **FULL** TABLES;

If we want to show or list the table name from different databases or database to which you are not connected without switching,

1. mysql> SHOW TABLES IN database\_name;

The above statement can also be written as:

1. mysql> SHOW TABLES **FROM** database\_name;
2. mysql> SHOW TABLES **FROM** sakila **WHERE** table\_type= "VIEW";

# MySQL Rename Table

1. mysql> RENAME old\_table **TO** new\_table;

We can also use the MySQL **RENAME TABLE** statement to change more than one table name with a single statement, as shown below:

RENAME **TABLE** old\_tab1 **TO** new\_tab1, old\_tab2 **TO** new\_tab2, old\_tab3 **TO** new\_tab3;

MySQL also use the RENAME TABLE statement for moving a table from one database to other database, which is show below:

1. mysql> RENAME **TABLE** current\_db.tablel\_name **TO** other\_db.tablel\_name;

# MySQL TRUNCATE Table

The TRUNCATE statement in MySQL removes the complete data without removing its structure.

The TRUNCATE command is more efficient as compared to the [DELETE](https://www.javatpoint.com/mysql-delete) command because it removes and recreates the table instead of deleting single records one at a time.

The following points must be considered while using the TRUNCATE command:

* We cannot use the **WHERE** clause with this command so that filtering of records is not possible.
* We **cannot rollback the deleted data** after executing this command because the log is not maintained while performing this operation.
* We cannot use the truncate statement when a table is referenced by a **foreign key** or participates in an **indexed view**.
* The TRUNCATE command doesn't fire DELETE **triggers** associated with the table that is being truncated because it does not operate on individual rows.

**TRUNCATE** [**TABLE**] table\_name;

The TABLE keyword in the syntax is not mandatory. But it's a good practice to use it to distinguish between the **TRUNCATE**() function and the **TRUNCATE TABLE statement**.

### How to Truncate Table with Foreign key?

If we perform the TRUNCATE operation for the table that uses a foreign key constraint, we will get the following error:

1. ERROR 1217 (23000): Cannot **delete** or **update** a parent row: a **foreign** **key** **constraint** fails

In that case, we need to log into the [MySQL](https://www.javatpoint.com/mysql-tutorial) server and **disable foreign key** checks before executing the TRUNCATE statement as below:

1. **SET** FOREIGN\_KEY\_CHECKS=0;

Now, we are able to truncate tables. After execution, **re-enable foreign key** checks as given below:

1. **SET** FOREIGN\_KEY\_CHECKS=1;

The TRUNCATE statement in MySQL will delete only one table at a time.

# MySQL DESCRIBE TABLE

Both DESCRIBE and DESC command are equivalent and case sensitive.

1. {DESCRIBE | **DESC**} table\_name;

### MySQL SHOW COLUMNS Command

1. mysql> SHOW COLUMNS **FROM** table\_name;

If we want to display the more column information, we need to add **FULL** keyword with the SHOW TABLES statement as follows:

1. mysql> SHOW **FULL** COLUMNS **FROM** table\_name;

# MySQL DROP Table

MYSQL uses a Drop Table statement to delete the existing table. This statement removes the complete data of a table along with the whole structure or definition permanently from the database.

1. mysql> **DROP** **TABLE** table\_name;
2. mysql> **DROP** **TABLE** schema\_name.table\_name;

# MySQL Copy/Clone/Duplicate Table

MySQL copy or clone table is a feature that allows us to create a ***duplicate table of an existing table***, including the table structure, indexes, constraints, default values, etc. Copying data of an existing table into a new table is very useful in a situation like backing up data in table failure.

1. **CREATE** **TABLE** IF NOT EXISTS new\_table\_name LIKE existing\_table\_name;
2. **INSERT** new\_table\_name **SELECT** \* **FROM** existing\_table\_name;
3. **CREATE** **TABLE** IF NOT EXISTS duplicate\_table **SELECT** \* **FROM** original\_table;

# MySQL Table Locking

A lock is a mechanism associated with a table used to restrict the unauthorized access of the data in a table. **MySQL allows a client session to acquire a table lock explicitly to cooperate with other sessions to access the table's data**. MySQL also allows table locking to prevent it from unauthorized modification into the same table during a specific period.

A session in MySQL can acquire or release locks on the table only for itself. Therefore, one session cannot acquire or release table locks for other sessions. It is to note that we must have a TABLE LOCK and SELECT privileges for table locking.

Table Locking in MySQL is mainly **used to solve concurrency problems**. It will be used while running a transaction, i.e., first read a value from a table (database) and then write it into the table (database).

[MySQL](https://www.javatpoint.com/mysql-tutorial) provides **two types of locks** onto the table, which are:

**READ LOCK:** This lock allows a user to only read the data from a table.

**WRITE LOCK:** This lock allows a user to do both reading and writing into a table.

The following is the syntax that allows us to acquire a table lock explicitly:

1. LOCK TABLES table\_name [**READ** | WRITE];

The following is the syntax that allows us **to release a lock** for a table in MySQL:

1. mysql> UNLOCK TABLES;

### Read vs. Write Lock

* Read lock is similar to "**shared**" locks because multiple threads can acquire it at the same time.
* Write lock is an "**exclusive**" locks because another thread cannot read it.
* We cannot provide read and write locks both on the table at the same time.
* Read lock has a **low priority** than Write lock, which ensures that updates are made as soon as possible.

# MySQL Constraints

The constraint in MySQL is used to specify the rule that allows or restricts what values/data will be stored in the table. It also helps to limit the type of data that will be inserted inside the table. If any interruption occurs between the constraint and data action, the action is failed.

### Types of MySQL Constraints

Constraints in MySQL is classified into two types:

1. **Column Level Constraints:** These constraints are applied only to the single column that limits the type of particular column data.
2. **Table Level Constraints:** These constraints are applied to the entire table that limits the type of data for the whole table.

### Constraints used in MySQL

The following are the most common constraints used in the MySQL:

* NOT NULL
* CHECK
* DEFAULT
* PRIMARY KEY
* AUTO\_INCREMENT
* UNIQUE
* INDEX
* ENUM
* FOREIGN KEY

# MySQL UPDATE Query

### The UPDATE statement is used with the **SET** and [**WHERE** clauses](https://www.javatpoint.com/mysql-where). The SET clause is used to change the values of the specified column.

### We can update single or multiple columns at a time.

1. **UPDATE** table\_name **SET** column\_name1 = new-value1,
2. column\_name2=new-value2, ... [**WHERE** Clause]

The UPDATE command supports these modifiers in MySQL:

**LOW\_PRIORITY:** This modifier instructs the statement to delay the UPDATE command's execution until no other clients reading from the table. It takes effects only for the storage engines that use only table-level locking.

**IGNORE:** This modifier allows the statement to do not abort the execution even if errors occurred. If it finds **duplicate-key** conflicts, the rows are not updated.

1. **UPDATE** Trainer\_table
2. **SET** email = REPLACE(email,'@javatpoint.com','@tutorialandexample.com')
3. **WHERE** course\_name = 'Testing';