**Database Export:**

The syntax for this statement combines a regular SELECT command with INTO OUTFILE filename at the end. The default output format is the same as it is for the LOAD DATA command. So, the following statement exports the tutorials\_tbl table into /tmp/tutorials.txt as a tab-delimited, linefeed-terminated file.

**select \* from Customers INTO OUTFILE "/tmp/customers.txt";**

You can change the output format using various options to indicate how to quote and delimit columns and records. To export the Customers table in a CSV format with CRLF-terminated lines, use the following code.

**select \* from Customers INTO OUTFILE "/tmp/customers4.txt" FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\r\n';**

Here, \r sent the print head back to the start of the line, and \n advanced the paper by one line.

Or, even better,

**select \* from Customers INTO OUTFILE "/tmp/customers4.csv" FIELDS TERMINATED BY ',' ENCLOSED BY '"' LINES TERMINATED BY '\r\n';**

**Now, SELECT …into OUTFILE has the following properties:**

* The output file is created directly by the MySQL server, so the filename should indicate where you want the file to be written on the server host. There is no LOCAL version of the statement analogous to the **LOCAL** version of **LOAD DATA**.
* You must have the **MySQL FILE** privilege to execute the **SELECT ... INTO** statement.
* The output file must not already exist. This prevents MySQL from clobbering files that may be important.
* You should have a login account on the server host or some way to retrieve the file from that host. Otherwise, the **SELECT ... INTO OUTFILE** command will most likely be of no value to you.
* Under UNIX, the file is created **world readable** and is owned by the MySQL server. This means that although you will be able to read the file, you may not be able to delete it.

Noted few weird things:

**select \* from Customers INTO OUTFILE "/tmp/customers.txt";**

Or,

**select \*from Customers INTO OUTFILE “customers.txt”;**

But,

**select \* from Customers INTO OUTFILE “/root/customers.txt” won’t work.**

## **Exporting Table Contents or Definitions in SQL Format:**

To export a table in SQL format to a file, use the command shown below.

**mysqldump -u root -p TUTORIALS tutorials\_tbl > dump.txt**

(if password is enabled/authentication is enabled

Otherwise, simply do,

**mysqldump -u root TUTORIALS tutorials\_tbl > dump.txt**

Now, this will write the following lines in the file:

-- MySQL dump 10.13 Distrib 5.1.73, for redhat-linux-gnu (x86\_64)

--

-- Host: localhost Database: trialDataBase8

-- ------------------------------------------------------

-- Server version 5.1.73

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `Customers`

--

DROP TABLE IF EXISTS `Customers`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `Customers` (

`CustomerID` int(11) NOT NULL AUTO\_INCREMENT,

`Name` varchar(80) NOT NULL,

`PhoneNumber` varchar(20) NOT NULL,

`Address` tinyblob NOT NULL,

`PostalCode` int(11) NOT NULL,

`COUNTRY` varchar(20) NOT NULL,

PRIMARY KEY (`CustomerID`)

) ENGINE=MyISAM AUTO\_INCREMENT=8 DEFAULT CHARSET=latin1;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `Customers`

LOCK TABLES `Customers` WRITE;

/\*!40000 ALTER TABLE `Customers` DISABLE KEYS \*/;

INSERT INTO `Customers` VALUES (1,'Alfreds Futterkiste','492180185611','Obere Str. 57, Berlin:-12209',12209,'Germany'),(2,'Sayak Haldar','9674465435','J2,102/B, DDA Flats, Kalkaji, Delhi-110019',110019,'India'),(3,'Sayantan Pandit','8697359734','Andul Mouri',711302,'India'),(4,'Suman Banerjee','9231837241','Andul PurboPara, Andul Mouri, Howrah',711302,'India'),(5,'Saptarshi Nag','8013362856','Bandra, West Mumbai',400050,'India'),(6,'Ankur Debnath','9734991000','j2,102/B, DDA Flats, Kalkaji, Delhi-110019',400050,'India'),(7,'Arinjoy Basak','5408381704','Blacksburg, VA 24061, USA',0,'USA');

/\*!40000 ALTER TABLE `Customers` ENABLE KEYS \*/;

UNLOCK TABLES;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2017-05-27 5:46:07

**To BackUp All The Databases Of Host:**

mysqldump -u root -p --all-databases > database\_dump.txt

(Note: mysqldump command is to be executed in

**DataBase Import (Recovery Methods):**

MySQL provides a LOAD DATA statement that acts as a bulk data loader. Here is an example statement that reads a file **dump.txt** from your current directory and loads it into the table **mytbl** in the current database.

**LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl;**

* If the **LOCAL** keyword is not present, MySQL looks for the datafile on the server host using the **looking into absolute pathname**, which fully specifies the location of the file, beginning from the root of the filesystem. MySQL reads the file from the given location.
* By default, **LOAD DATA** assumes that datafiles contain lines that are terminated by linefeeds (newlines) and that data values within a line are separated by tabs.
* To specify a file format explicitly, use a **FIELDS** clause to describe the characteristics of fields within a line, and a **LINES** clause to specify the line-ending sequence. The following **LOAD DATA** statement specifies that the datafile contains values separated by colons and lines terminated by carriage returns and new line character.

**LOAD DATA LOCAL INFILE 'dump.txt' INTO TABLE mytbl FIELDS TERMINATED BY ':’ LINES TERMINATED BY '\r\n';**

The LOAD DATA command assumes the columns in the datafile have the same order as the columns in the table. If that is not true, you can specify a list to indicate which table columns the datafile columns should be loaded into. Suppose your table has columns a, b, and c, but successive columns in the datafile correspond to columns b, c, and a.

## **Importing Data with mysqlimport:**

MySQL also includes a utility program named mysqlimport that acts as a wrapper around LOAD DATA, so that you can load the input files directly from the command line.

To load data from the dump.txt into mytbl, use the following command at the UNIX prompt.

mysqlimport -u root -p --local database\_name dump.txt

If you use mysqlimport, command-line options provide the format specifiers. The mysqlimport commands that correspond to the preceding two LOAD DATA statements looks as shown in the following code block.

mysqlimport -u root -p --local --fields-terminated-by = ":" \

--lines-terminated-by = "\r\n" database\_name dump.txt

The order in which you specify the options doesn't matter for mysqlimport, except that they should all precede the database name.

The mysqlimport statement uses the --columns option to specify the column order −

$ **mysqlimport -u root -p --local --columns=b,c,a \**

**database\_name dump.txt**

## **Handling Quotes and Special Characters In LoadData and MySQLImport:**

The FIELDS clause can specify other format options besides TERMINATED BY. By default, LOAD DATA assumes that values are unquoted and interprets the backslash (\) as an escape character for the special characters. To indicate the value quoting character explicitly, use the ENCLOSED BY command. MySQL will strip that character from the ends of data values during input processing. To change the default escape character, use ESCAPED BY.

When you specify ENCLOSED BY to indicate that quote characters should be stripped from data values, it is possible to include the quote character literally within data values by doubling it or by preceding it with the escape character.

For example, if the quote and escape characters are " and \, the input value "a""b\"c" will be interpreted as a"b"c.

For mysqlimport, the corresponding command-line options for specifying quote and escape values are --fields-enclosed-by and --fields-escaped-by.