1]- What is mysql ? [http://zetcode.com/databases/mysqltutorial/introduction/]

A- MySQL is a leading open source database management system. It is a multi-user, multithreaded database management system.

==> LAMP = Linux Apache Mysql PHP

==> The development of MySQL began in 1994 by a Swedish company MySQL AB. Sun Microsystems acquired MySQL AB in 2008.

==> Sun was bought by Oracle in 2010.

==> MySQL, PostgreSQL, Firebird, SQLite, Derby, and HSQLDB are the most well known open source database systems.

==> MySQL is developed in C/C++. Except for C/C++, APIs exist for PHP, Python, Java, C#, Eiffel, Ruby, Tcl or Perl.

**2]- Changing root password.**

$ mysql -u root -p

mysql > SET PASSWORD=PASSWORD('newpass');

*For Hardening*.

# mysql\_secure\_installation

**Setting up Test Database with test data**

========================================

# cd /var/lib/mysql ; wget http://downloads.mysql.com/docs/world.sql.gz

# gunzip world.sql.gz

# mysql -uroot -p

> create database world

> use world

> source world.sql

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

> create user user11@localhost IDENTIFIED BY 'user';

> grant all on world.\* to 'user11'@'localhost';

# mysql -uuser12 -p world -e "select \* from city limit 5"

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

# mysqldump -uroot -p world city > city.sql [backup of a single databas's table like city table of world database]

**3]- The mysqladmin tool**

# mysqladmin -uroot -p ping

# mysqladmin -uroot -p version

# mysqladmin -uroot -p status

# mysqladmin -uroot -p extended-status

**4]- Quick Access Notes**

======================

> select \* from city limit 5;

> select \* from city limit 15, 5;

> pager less [show output as less command]

> pager more [show output on screeen or disable less]

==>

> select count(ID) as 'ID\_Counts' from city;

> select \* from city where Population = (select max(Population) from city);

> select \* from city where Population = (select min(Population) from city);

> select \* from city where Name like 'Kal%';

> select \* from city where ID in (15,23,40);

> select \* from city where ID between 15 and 20;

**==> ORDERING**

> select \* from city ORDER BY ID desc limit 5 ;

> select \* from city ORDER BY ID asc limit 5 ;

**==> GROUPING (Aggregation function)** :- The GROUP BY clause is used to combine database records with identical values into a single record. It is often used with the aggregation functions.

> select Name,District,sum(Population) from city where District='New York' group by District;

> select Name,District,sum(Population) from city where District='New York' group by name;

> select District,sum(Population) from city where CountryCode = 'USA' group by District Having sum(Population) > 800000;

**==> Updating, Deleting**

> UPDATE country SET HeadOfState = 'Donald Trump' where Name = 'United State';

> Create table topten engine=memory select \* from country limit 15;

> delete from topten where id in (2,5,10);

> delete from topten where id=3;

> truncate table topten;

> insert into topten values(10,'Rohit','IN','Delhi',197079530);

> drop table topten;

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

**====== Storage =============================================**

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

==> \*Mysql Storage Engine\* :- A storage engine is a software module that a database management system uses to create, read, update data from a database. There are two types of storage engines in MySQL: transactional and non-transactional.

=> For MySQL 5.5 and later, the default storage engine is InnoDB.

=> The default storage engine for MySQL prior to version 5.5 was MyISAM.

MyISAM, InnoDB, Memory, CSV

=> MySQL supported storage engines:

1.InnoDB 2.MyISAM 3.Memory 4.CSV

5.Merge 6.Archive 7.Federated 8.Blackhole

**1.InnoDB :**- InnoDB is the most widely used storage engine with transaction support. It is an ACID compliant storage engine. It supports <row-level locking>, <crash recovery> and <multi-version concurrency control>. It is the only engine which provides foreign key referential integrity constraint. Oracle recommends using InnoDB for tables except for specialized use cases.

**2.MyISAM** :- MyISAM is the original storage engine. It is a fast storage engine. It does not support transactions. MyISAM provides table-level locking. It is used mostly in Web and data warehousing.

**3.Memory** :- Memory storage engine creates tables in memory. It is the fastest engine. It provides table-level locking. It does not support transactions. Memory storage engine is ideal for creating temporary tables or quick lookups. The data is lost when the database is restarted.

**4.CSV** :- CSV stores data in CSV files. It provides great flexibility because data in this format is easily integrated into other applications.

**5.Merge** :- Merge operates on underlying MyISAM tables. Merge tables help manage large volumes of data more easily. It logically groups a series of identical MyISAM tables, and references them as one object. Good for data warehousing environments.

**6.Archive** :- Archive storage engine is optimised for high speed inserting. It compresses data as it is inserted. It does not support transactions. It is ideal for storing and retrieving large amounts of seldom referenced historical, archived data.

**7.Blackhole** :- The Blackhole storage engine accepts but does not store data. Retrievals always return an empty set. The functionality can be used in distributed database design where data is automatically replicated, but not stored locally. This storage engine can be used to perform performance tests or other testing.

**8.Federated** :- Federated storage engine offers the ability to separate MySQL servers to create one logical database from many physical servers. Queries on the local server are automatically executed on the remote (federated) tables. No data is stored on the local tables. It is good for distributed environments.

> show engines\G

> show engines;

> CREATE TABLE Cars(Id INTEGER PRIMARY KEY, Name VARCHAR(50),Cost INTEGER) ENGINE='MyISAM';

> select TABLE\_SCHEMA,TABLE\_NAME,ENGINE from information\_schema.TABLES group by ENGINE;

> select TABLE\_SCHEMA,TABLE\_NAME,ENGINE from information\_schema.TABLES where TABLE\_SCHEMA='world' and TABLE\_NAME='city';

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

===== MySQL data types ============================================

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

Data Types :- A data type is a set of representable values. Each value belongs to one data type. Items that can be referenced by a name, such as SQL parameters, columns, fields, attributes, and variables, also have declared types.

**Groups of Data types :-**

**1- Numeric 2- Data & Time 3- String 4- JSON**

1- Integers ( TINYINT,SMALLINT, MEDIUMINT, INTEGER, BIGINT)

Floating points (FLOAT, DOUBLE, DECIMAL)

(Floating point number represent real numbers in computing.Real numbers measure continuous quantities, like weight, height or speed)

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

Data type Bytes Minimum value Maximum value

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

TINYINT 1 -128 127

SMALLINT 2 -32768 32767

MEDIUMINT 3 -8388608 8388607

INTEGER 4 -2147483648 2147483647

BIGINT 8 -9223372036854775808 9223372036854775807

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

Data type Bytes Minimum value Maximum value

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

TINYINT 1 0 255

SMALLINT 2 0 65535

MEDIUMINT 3 0 16777215

INTEGER 4 0 4294967295

BIGINT 8 0 18446744073709551615

mysql> CREATE TABLE Ages(Id SMALLINT, Age TINYINT) ENGINE=Memory;

MariaDB [world]> select 1/3;

+--------+

| 1/3 |

+--------+

| 0.3333 |

+--------+

> CREATE TABLE Numbers (Id TINYINT, Floats FLOAT, Decimals DECIMAL(3, 2));

> INSERT INTO Numbers VALUES (1, 1.1, 1.1), (2, 1.1, 1.1), (3, 1.1, 1.1);

> SELECT SUM(Floats), SUM(Decimals) FROM Numbers;

**==== Date & Time ==============================================**

http://zetcode.com/databases/mysqltutorial/datatypes/

> select TABLE\_SCHEMA,TABLE\_NAME,ENGINE from information\_schema.TABLES where TABLE\_SCHEMA='world';

> create table Dates(ID tinyint,Dates date);

> INSERT INTO Dates VALUES(1, '2017-01-24');

> INSERT INTO Dates VALUES(2, '2017/01/25');

> INSERT INTO Dates VALUES(3, '20170126');

> INSERT INTO Dates VALUES(4, '170127');

> INSERT INTO Dates VALUES(5, '2017+01+28');

> INSERT INTO Dates VALUES(6, '17+01+27');

> select curdate()

> select curtime()

> select now();

> select date(now());

> select year(now());

> select time(now());

> select adddate(date(now()),8);

> select adddate('2019-03-21',8);

> select addtime('10:25:05','1:00:00');

> select addtime(time(now()),'-1:00:00');

> select timediff('11:25:05','10:25:05');

> select datediff('2019-04-21','2019-03-21');

> select dayname(now());

> select daytime('2019-03-21 11:36:06');

**===== Timestamp ===============================================**

=> A timestamp is a sequence of characters, denoting the date and/or time at which a certain event occurred.

=> Timestamps are typically used for logging events

=> A TIMESTAMP column is useful for recording the date and time of an INSERT or UPDATE operation.

=> It automatically sets to the date and time of the most recent operation if do not give it a value yourself.

=> The TIMESTAMP data type has a range of 1970-01-01 00:00:01 UTC to 2038-01-19 03:14:07 UTC.

=> The TIMESTAMP data type offers automatic initialisation and updating.

=> We can restrict this data type to have only automatic initialisation or automatic update only

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

Data type Format

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

TIMESTAMP(14) YYYYMMDDHHMMSS

TIMESTAMP(12) YYMMDDHHMMSS

TIMESTAMP(10) YYMMDDHHMM

TIMESTAMP(8) YYYYMMDD

TIMESTAMP(6) YYMMDD

TIMESTAMP(4) YYMM

TIMESTAMP(2) YY

> create table price (ID tinyint primary key,Price float,Stamps timestamp) engine=InnoDB;

> alter table price modify Price decimal(10,2);

> insert into price(ID,Price) values(1,101.35);

> update table price set Price=105.23 where ID=1;

**==== Strings =================================================**

MySQL has the following string data types:

1.CHAR 2.VARCHAR 3.BINARY 4.VARBINARY 5.BLOB 6.TEXT 7.ENUM 8.SET

**1.CHAR :-**

=> A CHAR is a fixed length character data type. It is declared with a length, CHAR(x), where x can be between 0 to 255.

=> CHAR always uses the same amount of storage space per entry.

=> In case we specify an item which is shorter than the declared length, the value is right-padded with spaces to the specified

length. Trailing spaces are removed when the value is retrieved.

> create table Chars(ID smallint primary key,name char(5));

> insert into Chars values (101,'ROHIT'),(102,'a'),(103,'bb');

> select ID,length(name) as Length from Chars;

**2.VARCHAR :-**

=> VARCHAR data types stores variable-length strings. The length of the string can be from 0 to 65535.

=> VARCHAR values are not padded when they are stored. Trailing spaces are retained when values are stored and retrieved.

=> Most shorter string data types are stored in this data type; for example emails, names of people, of merchandise, or addresses.

> CREATE TABLE FirstNames(Id TINYINT, Firstname VARCHAR(20));

> INSERT INTO FirstNames VALUES (1,'Tom'),(2,'Lucy'),(3,'Alice'),(4,'Robert'),(5,'Timothy'),(6,'Alexander');

**3.BINARY :-**

=]> BINARY and VARBINARY are binary byte data types. They contain byte strings rather than character strings.

=]> They have no character sets. Sorting and comparison are based on the numeric values of the bytes in the values.

=]> The range of the BINARY data types is from 0 to 255. It stores values in fixed length.

=]> The range of the VARBINARY is from 0 to 65535.

**5.BLOB :-** [https://dev.mysql.com/doc/refman/5.7/en/load-data.html]

=> A BLOB is a binary large object data type.

=> It can hold a variable amount of binary data.

=> It can be used to store binary data like images or documents.

BLOB has four types :-

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

Blog type Range in bytes

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

TINYBLOB 0 - 255

BLOB 0 - 65535

MEDIUMBLOB 0 - 16777215

LONGBLOB 0 - 4294967295

> create table images (ID int primary key,Img longblob);

> show VARIABLES LIKE "secure\_file\_priv";

> insert into images values(1,load\_file('/var/lib/mysql-files/img.png'));

> select img from images where ID=1 into dumpfile '/var/lib/mysql-files/img-bak.png';

**6. TEXT :-**

=> A TEXT datatype is used for storing large textual data.

=> Example articles, blogs, or pages.

=> There is no padding on insert and no bytes are stripped on select.

=> TEXT values are best used when VARCHAR and other string-based data objects are insufficient to handle storing the desired amount of information.

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

Blog type Range in bytes

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

TINYTEXT 0 - 255

TEXT 0 - 65535

MEDIUMTEXT 0 - 16777215

LONGTEXT 0 - 4294967295

**7.ENUM :-**

=> The ENUM is a string object with a value chosen from a permitted list of values.

=> They are enumerated explicitly in the column specification.

=> We can insert only one value from the list.

> CREATE TABLE Sizes(Size ENUM('S', 'M', 'L', 'XL', 'XXL'));

> insert into sizes values('S'),('M');

> insert into sizes values('Y'); [Will throw warning,& not insert.]

> show warnings;

**8.SET** **:-**

=> A SET is a string object that can have zero or more values, each of which must be chosen from a list of permitted values.

=> It is similar to the ENUM data type.

=> The difference is that it can contain zero or more values from the list of permitted values.

mysql> CREATE TABLE Letters(Let SET('a', 'b', 'c', 'd', 'e'));

We create a table that allows a set of letters on a column.

mysql> INSERT INTO Letters VALUES ('a');

mysql> INSERT INTO Letters VALUES ('b');

mysql> INSERT INTO Letters VALUES ('b,a');

mysql> INSERT INTO Letters VALUES ('');

mysql> INSERT INTO Letters VALUES ('a,b,c');

**==== JSON (JavaScript Object Notation) ==============================**

=> since version 5.7.8, mysql support a native JSON data types

=> JSON (JavaScript Object Notation) is a lightweight data-interchange format.

=> It is easy for humans to read and write and for machines to parse and generate.

=> MySQL automaticcally validates JSON documents stored in JSON columns. Invalid documents produce an error.

=> JSON documents stored in JSON columns are optimized for efficient access.

=> JSON columns cannot have a default value.

$ mysqladmin -V

$ mysql -uroot -p -e "select version();"

$ mysql -uroot -p -e "show VARIABLES like '%version%';"

=> The JSON\_ARRAY() function takes a list of values and transforms them into a JSON array.

=> The JSON\_OBJECT() function takes a list of key/value pairs a nd returns a JSON object containing those pairs.

> CREATE TABLE t1 (Doc JSON);

> INSERT INTO t1 VALUES('{"chair": "5", "table": "4", "lamp": "6"}');

> select \* from t1;

> SELECT JSON\_ARRAY('pen', 4, 'pencil', 2, 'rubber', 1);

> SELECT JSON\_OBJECT('pen', 4, 'pencil', 2, 'rubber', 1);

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

==== CREATE, ALTER and DROP tables in DATABASE===========================

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

> create table testing(ID tinyint) eninge=InnoDB;

> show create table testing;

> show tables like 't%';

> drop table testing;

> alter table testing rename test;

> alter table test modify ID varchar(20);

> show create table test;

> create table t1 (ID int not null) character set='utf8' collat='utf8\_general\_ci';

> alter table t1 add field1 varchar(10) not null;

> alter table t1 drop field1 ;

> show columns from t1;

> describe t1;

[https://dev.mysql.com/doc/refman/5.7/en/charset-applications.html]

[https://dev.mysql.com/doc/refman/8.0/en/adding-collation.html]

[http://www.unicode.org/reports/tr10/]

====== Character set collat ==============

=> A collation is a set of rules that defines how to compare and sort character strings

=> A collation orders characters based on weights.

=> Each collation in MySQL belongs to a single character set.

=> Every character set has at least one collation, and most have two or more collations.

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

==== MySQL Expressions ==========================================

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

An expression in a programming language is a combination of values, variables, operators, and functions that are interpreted (evaluated) according to the particular rules of precedence and of association for a particular programming language, which computes and then produces (returns, in a stateful environment) another value. The expression is said to evaluate to that value.

**1. Literal Values :-**

=> A literal value is a constant of some kind. Literal values may be strings, numbers, hexadecimal values, boolean values, and NULL.

> select 3,'wolf',0x34,0+b'011';

> select NULL, \N ;

> select TRUE,FALSE;

**2. Variables :**-

=> A variable is a symbolic name associated with a value. This value may be changed over time.

=> Variables in MySQL are preceded by the @ character.

> set @name = 'Rohit';

> select @name ;

**3.Operators :**-

=> Operators are used to build expressions. SQL operators are very similar to mathematical operators.

=> There are two kinds of operators. Binary and unary.

\* Binary operators work with two operands, unary work with one.

=> An operator may have one or two operands. An operand is one of the inputs (arguments) of an operator.

=> We have several types of operators:

1. Arithmetic operators(+, -, /, \*) 2.Boolean operators( and, or, xor) 3.Relational operators (use to compare value)

4.Bitwise operators 5.Other operators

> select -(3-4);

> select (3+4);

> select not (3<9);

> select (3<9);

> select 5/4,5 div 4;

> select false and false, true and true ;

**5. Other Operator :-**

> select 'i' in ('a','i','z');

> set @run = true

> select @run is true;

> select \* from city where name like 'Hon%';

> select \* from city where name like '\_\_\_'; {Three \_ underscores}

> select \* from city where Name regexp 'ery.$';

(between)

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

==== Data Manipulation ===========================================

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

> CREATE TABLE Books2(Id INTEGER PRIMARY KEY AUTO\_INCREMENT, Title VARCHAR(100), Author VARCHAR(60)) type=MEMORY;

> INSERT INTO Books(Title, Author) VALUES ('The Insulted','Fyodor Dostoyevsky'), ('Cousin Bette', 'Honore de Balzac');

**==== Insert data from Table 2 Table ===================================**

> CREATE TABLE Books2(Id INTEGER PRIMARY KEY AUTO\_INCREMENT,Title VARCHAR(100), Author VARCHAR(60)) type=MEMORY;

> insert into Books2 select \* from books;

**==== Working with CSV files =========================================**

**[EXPORT]**

> select \* into OUTFILE '/tmp/file1.csv' FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' from city;

> truncate city

**[IMPORT]**

> LOAD DATA INFILE '/tmp/file1.csv' INTO TABLE book FIELDS TERMINATED BY ',' LINES SEPERATED BY '\n';

# ls /tmp/systemd-private-d195696fe51849f09a8d0e440b3ed431-mariadb.service-22CgMF/tmp/

books2.csv books.csv file1.csv file2.csv

**==== Working with XML File =========================================**

**[EXPORT]**

> mysql -uroot -p --xml -e 'select \* from world.book' > bookbak.xml

> mysql -uroot -p --xml -e 'truncate world.book'

**[IMPORT]**

> mysql -uroot -p -e "LOAD XML INFILE '/tmp/bookbak.xml' INTO TABLE world.book;"

[/tmp/systemd-private-d195696fe51849f09a8d0e440b3ed431-mariadb.service-22CgMF/tmp/bookbak.xml]

> mysql -uroot -p -e "select count(\*) from world.book;"