MYSQL Tutorial: Basics to Advanced

1. MYSQL COMMAND LINE COMMANDS

COMMAND	MEANING	SYNTAX
mysql	Allows user to connect to the MySQL CLI	`>MYSQL -U [USERNAME] -P;`
exit	Exits the MySQL CLI	`>EXIT;`
clear	Clears the MySQL shell	`>SYSTEM CLEAR;`
create user	Creates a new user	`` >CREATE USER `NEWUSER`@`LOCALHOST` IDENTIFIED BY `NEW_PASSWORD` ``
show user	Shows all user who have access to the MySQL Client	`>SELECT USER, HOST FROM MYSQL.USER;`
drop user	To delete an existing user	`> DROP USER 'USERNAME'@'LOCALHOST';`
grant all privileges	Assigns privileges to a MySQL user	`>GRANT ALL PRIVILEGES ON * . * TO 'USERNAME'@'LOCALHOST';`
show grants	Displays the privileges that are assigned to a MySQL user	`> SHOW GRANTS FOR 'USERNAME'@'LOCALHOST';`
revoke all privileges	Revokes all privileges assigned to a MySQL user	`>REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'USERNAME'@'LOCALHOST';`
mysqldump	Creates a backup of a set of SQL statements that can be used to recreate the original database object definitions and table data.	`>MYSQLDUMP -U USERNAME -P DATABASENAME> DATABASENAME_BACKUP.SQL`

2. MYSQL DATABASE COMMANDS (DATA DEFINITION LANGUAGE;DDL)

COMMAND	MEANING	SYNTAX
show database	Shows all the databases available in MySQL server.	`>SHOW DATABASE;`
create database	Creates a new database if it does not exist.	`>CREATE DATABASE DATABASENAME;`
drop database	To delete an existing database permanently.	`>DROP DATABASE  DATABASE_NAME`
alter database	Changes or modifies the characteristics of an existing database.	`>ALTER DATABASE [DATABASENAME] ALTEROPTION ;`
use database	Allow you to use a particular database or change from the current database to another database.	`>USE DATABASENAME;`

3. MySQL Table commands(DDL)

COMMAND	MEANING	SYNTAX
show tables	Shows all tables within the current database.	`>SHOW TABLES;`

create table	Creates a new table in the current database.	`>CREATE TABLE TABLENAME (`
		`    COLUMN1 DATATYPE,`
		`    COLUMN2 DATATYPE,`
		`    COLUMN3 DATATYPE,`
		`   ....`
		`CONSTRAINTS ....`
		`);`

alter table (add column)	Adds a new column to an existing table.	`>ALTER TABLE TABLENAME ADD COLUMNNAME DATATYPE;`
alter table (drop column)	Deletes a column from an existing table.	`>ALTER TABLE TABLENAME DROP COLUMN COLUMNNAME;`

alter table (alter column)	Alters an existing column in an already existing table.	`>ALTER TABLE TABLENAME`
		`ALTER COLUMN COLUMNNAME DATATYPE;`

alter table(add primary key)	Alters or adds primary key to an existing table.	`>ALTER TABLE TABLENAME`
		`ADD PRIMARY KEY (COLUMNNAME,...);`

COMMAND	MEANING	SYNTAX
alter table(drop primary key)	Drops an existing primary key in a table.	>ALTER TABLE TABLENAME DROP PRIMARY KEY;
alter table(add foreign key)	Creates a foreign key on an existing table.	>ALTER TABLE TABLENAME1 ADD FOREIGN KEY (COLUMN1) REFERENCES TABLENAME2(COLUMN2);
alter table(drop foreign key)	Deletes an existing foreign key in an already existing table.	> ALTER TABLE TABLENAME DROP FOREIGN KEY FOREIGNKEY_NAME;
rename table	Changes the name of an existing table.	>RENAME TABLE OLD_TABLENAME TO NEW_TABLENAME;
drop table	Deletes the entire table along with its definition.	>DROP TABLE TABLE_NAME;
truncate table	Remove all records in a MySQL table.	>TRUNCATE TABLE TABLENAME;
describe table	Displays all the columns of an existing table.	>DESCRIBE TABLE_NAME;
describe table column	Displays all the values stored in a particular column.	>DESCRIBE TABLE_NAME COLUMN_NAME;

---

4. MySQL DML(Data Manipulation Language) Commands

COMMAND	MEANING	SYNTAX
select *	Displays all rows in a table.	>SELECT * FROM TABLENAME
select * (multiple tables)	Displays all the rows of the cartesian product of the two tables	>SELECT * FROM TABLENAME1,TABLENAME2;
select columns	Select particular columns from table(s)	>SELECT COLUMN1,COLUMN2 FROM TABLENAME;
select with condition	Displays rows based on a particular condition	> SELECT * FROM TABLENAME WHERE CONDITION
select with multiple conditions(AND)	Displays rows only when both the conditions are satisfied.	> SELECT * FROM TABLENAME WHERE CONDITION1 AND CONDITION2.
select with multiple conditions(OR)	Displays rows only when either of the conditions are satisfied.	> SELECT * FROM TABLENAME WHERE CONDITION1 OR CONDITION2.
select with condition(NOT)	Displays rows based on negation of a particular condition.	>SELECT * FROM TABLENAME WHERE NOT CONDITION.
select with group by	Displays rows that have same values into summary rows	> SELECT .. FROM .. WHERE… GROUP BY COLUMN3;
select with having	Used instead of where for aggregate functions.	>SELECT COUNT(COLUMN1) FROM TABLENAME ORDER BY COLUMN2 HAVING COUNT(COLUMN1)>3;
select distinct	Display all unique rows discarding duplicate ones.	>SELECT DISTINCT (COLUMN1) FROM TABLENAME;
order by	Used to sort results in ascending order or descending order	> SELECT … FROM TABLENAME ORDER BY COLUMN1 ASC|DESC;
column alias	Changes the output of the name of the column.	> SELECT COLUMN1 AS NEWNAME FROM TABLENAME;
like	Used to search for a specific pattern.	> SELECT COLUMN1 FROM TABLENAME WHERE COLUMN1 LIKE '%PATTERN%';
insert record	Adds a new row to an existing table.	> INSERT INTO TABLENAME (COLUMN1,COLUMN2…) VALUES (VALUE1,VALUE2…);
insert record(multiple)	Adds multiple records into an existing table.	> INSERT INTO TABLENAME (COLUMN1,COLUMN2…) VALUES (VALUE1A,VALUE2A…),(VALUE1B,VALUE2B,...);
delete	Deletes all records in a table.	> DELETE FROM TABLENAME;

COMMAND	MEANING	SYNTAX
delete with where	Deletes specific records	`>DELETE FROM TABLENAME WHERE CONDITION;`
between	Selects values in a given range	`>SELECT * FROM TABLENAME WHERE AGE BETWEEN 25 AND 30.`
in	Used instead of multiple OR operators.	`> SELECT * FROM TABLENAME WHERE COLUMN2 IN (V1,V2…);`
exists	Tests for existence of a certain record. Returns a boolean value.	`> SELECT * FROM TABLE NAME WHERE EXIST (SUB QUERY);`
update table	Modifies data in existing tables.	`> UPDATE TABLENAME SET COLUMNNAME=VALUE WHERE CONDITION;`

inner join	Selects records that have the same values in two same or distinct tables.	`> SELECT COLUMN(S) FROM TABLENAME1 INNER JOIN TABLENAME2`
		`ON TABLENAME1.COLUMNAME=TABLENAME2.COLUMNAME;`

left join	Selects all the records from the left table and matching records from the right table.	`>SELECT COLUMN(S) FROM TABLENAME1 LEFT JOIN TABLENAME2`
		`ON TABLENAME1.COLUMNAME=TABLENAME2.COLUMNAME;`

right join	Selects all the records from the right table and matching records from the left table.	`>SELECT COLUMN(S) FROM TABLENAME1 RIGHT JOIN TABLENAME2`
		`ON TABLENAME1.COLUMNAME=TABLENAME2.COLUMNAME;`

cross join	Selects rows from cartesian product of both the tables.	`>SELECT COLUMN(S) FROM TABLE1 CROSS JOIN TABLE2;`

full outer join	Selects all records with a match on table1 or table2.	`>SELECT COLUMN(S) FROM TABLENAME1 FULL OUTER JOIN TABLENAME2`
		`ON TABLENAME1.COLUMNAME=TABLENAME2.COLUMNAME WHERE CONDITION;`

union	Combines the result of two select statements.	`>SELECT * FROM TABLENAME1`
		`UNION SELECT * FROM TABLENAME2`

union all	Similar to Union but allows duplicate values	`>SELECT * FROM TABLENAME1`
		`UNION ALL SELECT * FROM TABLENAME2`

concat()	Combines two or more columns together.	`>SELECT CONCAT(COLUMN1, " ", POSTALCODE, " ", COLUMN2)`
		`AS NEWCOL`
		`FROM TABLENAME;`

5. MySQL DATA TYPES

In MySQL just like other programming languages, each column, local variable, expression, and parameter has a related data type. A data type is an attribute that specifies the type of data that the object can hold.

- **String Data Types**

DATATYPE	DETAILS
CHAR(size)	Stores Alpha Numeric and special characters. Size varies from 0 to 255 characters.
VARCHAR(size)	Can contain letters, numbers, and characters that are of variable length (size). The size parameter specifies the column length in characters; it can be from 0 to 65535.
BINARY(size)	Similar to CHAR(). But it stores binary strings.
VARBINARY(size)	Similar to Binary() but the length is variable.
TINYBLOB	For Binary Large Objects. Max size=255 bytes.
TINYTEXT	Holds string of max length 255 characters.
TEXT(Size)	Stores a string of max length 65535 bytes.
BLOB	Stores Binary Large Objects up to 65535 bytes of data.
MEDIUMTEXT	Stores 2^8 times the characters as compared to TINYTEXT.
MEDIUMBLOB	Stores 2^8 times bytes as compared to TINYBLOB.
LONGTEXT	Stores 2^8 times the characters as compared to MEDIUMTEXT.
LONGBLOB	Stores 2^8 times bytes as compared to MEDIUMBLOB.
ENUM(val1, val2, val3, ...)	Stores only one value, which can be chosen from a range of possible values. An ENUM list can contain at most 65535 values. A value that is inserted that is not in the list will be replaced with a blank value. The values are arranged in the order you specify them.
SET(val1, val2, val3, ...)	Stores a string object that can have 0 or more values, chosen from a list of possible values. You can list up to 64 values in a SET list

- **Numeric Data Types**

DATATYPE	DETAILS
BIT(size)	Stores a bit-value. The size parameter specifies the number of bits per value . The value is represented as a number of bits. The size parameter can hold a value from 1 to 64. The default value for size is 1.
TINYINT(size)	Stores very small int values. Signed ranges from -128 to 127. Unsigned ranges from 0 to 255. Size defines the maximum display width of 255.
BOOL	Zero is considered as false and one is considered as true.
BOOLEAN	Same as BOOL.
SMALLINT(size)	Stores a small integer. Signed ranges from -32768 to 32767. Unsigned ranges from 0 to 65535. Size defines the maximum display width of 255.

DATATYPE	DETAILS
MEDIUMINT(size)	Stores a medium valued integer. Signed ranges from -8388608 to 8388607. Unsigned ranges from 0 to 16777215. Size defines the maximum display width of 255.
INT(size)	Stores a medium integer. Signed ranges from -2147483648 to 2147483647. Unsigned ranges from 0 to 4294967295. Size defines the maximum display width of 255.
INTEGER(size)	Same as INT(size)
BIGINT(size)	Stores a large valued integer. Signed ranges from -9223372036854775808 to 9223372036854775807. Unsigned ranges from 0 to 18446744073709551615. Size defines the maximum display width of 255.
FLOAT(size, d)	Stores a floating point(decimal number). The number of digits is specified in size. The number of digits after the decimal point is specified by the value d.
FLOAT(p)	Stores a floating point(decimal number. If p value is between 0 and 24, the data type becomes FLOAT() else the data type becomes DOUBLE()
DOUBLE(size, d)	Stores a normal-size floating point (decimal)number. The number of digits is specified in size. The number of digits after the decimal point is specified by the value d.
DECIMAL(size, d)	An exact fixed-point number. The total number of digits is specified in size. The number of digits after the decimal point is specified in the d parameter. The maximum number for size is 65. The maximum number for d is 30. The default value for size is 10. The default value for d is 0.

- **Date and Time Data Types**

DATATYPE	DETAILS
DATE	Stores a date in the format: YYYY-MM-DD. Supports a range between '1000-01-01' to '9999-12-31'
DATETIME(fsp)	Combination of date and time in the format: YYYY-MM-DD hh:mm:ss. Supports a range between '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
TIMESTAMP(fsp)	Stores a time stamp in the format YYYY-MM-DD hh:mm:ss UTC. Supports a range between '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	Stores time in the format hh:mm:ss. Supports a range between '-838:59:59' to '838:59:59'
YEAR	Stores a year in four-digit format. Supports a range between 1901 to 2155 (includes 0000).

6. MySQL AGGREGATE FUNCTIONS

A function that performs an arithmetic operation on a set of values and returns a single value is called an aggregate function.

COMMAND	FUNCTION	SYNTAX

count()	Returns the number of rows, (including NULL)	`>SELECT COUNT(COLUMN_NAME)`
		`FROM TABLE_NAME`
		`WHERE CONDITION;`

sum()	Returns sum of all non NULL values.	`>SELECT SUM(COLUMN_NAME)`
		`FROM TABLE_NAME`
		`WHERE CONDITION;`

avg()	Returns average of all non NULL values.	`>SELECT AVG(COLUMN_NAME)`
		`FROM TABLE_NAME`
		`WHERE CONDITION;`

min()	Returns minimum value in the set.	`>SELECT MIN(COLUMN_NAME)`
		`FROM TABLE_NAME`
		`WHERE CONDITION;`

max()	Returns maximum value in the set.	`>SELECT MAX(COLUMN_NAME)`
		`FROM TABLE_NAME`
		`WHERE CONDITION;`

groutp_concat()	Concatenates values from multiple rows into one field.	`>SELECT COLUMN1, COLUMN2, ...`
		`GROUP_CONCAT ( DISTINCTCOLUMN1`
		`ORDER BY ..   )`
		`FROM TABLE_NAME GROUP BY COLUMN2;`

7. INDEXES AND VIEWS IN MySQL

An Index retrieves data much faster than otherwise. Indexes speed up the query/search. A user cannot view an Index. Updating a table with an index takes more time because both table and index have to be updated.
The view is a virtual table which takes the result of an SQL query. Users can access a View. They have rows and columns similar to a table.

COMMAND	FUNCTION	SYNTAX

create index	Creates a new index from an existing table. Allows duplicate values.	`> CREATE INDEX indexname`
		`ON tablename (column1, column2, ...);`

create index unique	Similar to creating an index. But only allows unique values.	`>CREATE UNIQUE INDEX indexname`
		`ON tablename (column1, column2, ...);`

drop index	Deletes an existing index.	`> DROP INDEX INDEXNAME;`
rebuild index	Used to rebuild one or all indexes in a table if corrupted.	`>REINDEX INDEX INDEXNAME;`

create view	Creates a view if it doesn’t exist.	`> CREATE VIEW VIEWNAME AS SELECT COLUMN1,COLUMN2 FROM`
		`TABLE WHERE CONDITION;`

COMMAND	FUNCTION	SYNTAX

update view	Creates or edits an existing view.	`> CREATE OR REPLACE viewname`
		`AS`
		`SELECT COLUMN1,COLUMN2 FROM TABLE WHERE CONDITION;`

rename view	Changes the name of the view.	`> RENAME TABLE VIEWNAME TO NEWVIEWNAME;`
drop view	Deletes an existing view.	`> DROP VIEW VIEWNAME;`
drop views	Deletes multiple views.	`> DROP VIEW VIEW1,VIEW2…;`

show views	Displays all views in a database.	`> SHOW FULL TABLES`
		`[{FROM | IN } databasename]`
		`WHERE table_type = 'VIEW';`

8. TRIGGERS IN MYSQL

Triggers are DBMS objects which are associated with tables. Triggers are fired when any one of the DML statements (INSERT, DELETE or UPDATE) is activated.
There are two types of triggers,

- Row Level Triggers: A trigger is an instruction that causes a row to trigger to be fired once for each row affected by an insert, update, or delete statement. The row trigger is fired automatically.
- Statement Level Trigger: Trigger is fired once regardless of the number of DML statements.

There are six types of triggers, namely,

- Before Insert: Activated before insertion.
- After Insert: Activated after insertion.
- Before Update: Activated before updating.
- After Update: Activated after updating.
- Before Delete: Activated before deletion.
- After Delete: Activated after deletion.

COMMAND	FUNCTION	SYNTAX

create trigger	Creates a new trigger on an existing table.	`>CREATE TRIGGER TRIGGERNAME`
		`BEFORE | AFTER INSERT | UPDATE| DELETE`
		`ON TABLENAME FOR EACH ROW`
		`TRIGGERBODY;`

drop trigger	Deletes an existing trigger.	`> DROP TRIGGER TRIGGERNAME;`
show all triggers	Displays all the triggers in the database.	`> SHOW TRIGGERS FROM | IN DATABASE_NAME WHERE SEARCH_CONDITION;`

9. STORED PROCEDURES AND FUNCTION

Procedures are reusable SQL codes that we store in a database. We can directly call procedures instead of writing the query again and again.
Functions are reusable code, which runs certain SQL commands and returns an appropriate value.

- **Syntax to create a new procedure.**

```
DELIMITER $$
CREATE PROCEDURE procedurename(parameterlist)
BEGIN
  body;
END $$
DELIMITER ;
```

- **Syntax to create a new function**

```
DELIMITER $$
CREATE FUNCTION functionname(parameterlist)
RETURNS datatype
NOT DETERMINISTIC
BEGIN
 %statements%
END $$

DELIMITER ;
```

COMMAND	FUNCTION	SYNTAX
drop procedure	Deletes an existing procedure.	`> DROP PROCEDURE PROCEDURENAME;`
show all procedures	Displays all the stored procedures in the database.	`> SHOW PROCEDURE STATUS LIKE '%PATTERN' | WHERE CONDITION;`
drop function	Deletes an existing stored function.	`> DROP FUNCTION FUNCTIONNAME;`
show stored functions	Displays all the stored functions.	`> SHOW FUNCTION STATUS LIKE '%PATTERN' | WHERE CONDITION;`

10. INBUILT FUNCTIONS IN MySQL

- **STRING FUNCTIONS**

Function	Description
ASCII	Returns the ASCII value of a character
CHAR_LENGTH	Returns the length of a string.
CHARACTER_LENGTH	Returns the length of a string
CONCAT	Concatenates two or more expressions.
CONCAT_WS	Concatenates with a separator.
FIELD	Returns the index of value in a list.

Function	Description
FIND_IN_SET	Returns the index of a string within a list.
FORMAT	Changes the format/representation.
INSERT	Inserts a string within a string at a given index.
INSTR	Returns the index of the first occurrence of a string in another one.
LCASE	Converts an entire string to lowercase.
LEFT	Extracts a length of characters from the left of a string.
LENGTH	Returns the string length in bytes.
LOCATE	Returns the location of the first occurrence of a substring in a given string
LOWER	Converts an entire string to lowercase.
LPAD	Left-pads a string with a given string.
LTRIM	Removes spaces from the left of a string.
MID	Extracts a substring from a string at a given position.
POSITION	Returns the location of the first occurrence of a substring in a given string
REPEAT	Repeats the string the number of times the user specifies.
REPLACE	Replaces occurrences of a substring in a string with another substring.
REVERSE	Reverses the string.
RIGHT	Extracts a length of characters from the right of a string.
RPAD	Right-pads a string with a given string.
RTRIM	Removes spaces from the right of a string.
STRCMP	Checks whether two strings are equal.
SUBSTR	Extracts a substring from a string at a position mentioned by the user.
SUBSTRING	Same as substr.
TRIM	Trims leading and trailing spaces from a string as specified by the user.
UCASE	Converts an entire string to uppercase.
UPPER	Converts an entire string to uppercase.

- **NUMERIC FUNCTIONS**

Function	Description
ABS	Returns the absolute value.
ACOS	Returns the cosine inverse.
ASIN	Returns the sine inverse.
ATAN	Returns the tan inverse of one or two numbers.
ATAN2	Returns the tan inverse of two numbers.
AVG	Returns the mean value.
CEIL	Returns the smallest integer that is greater than or equal to the number
CEILING	Returns the smallest integer that is greater than or equal to the number
COS	Returns the cosine.
COT	Returns the cotangent.
COUNT	Returns the number of records returned by a query.
DEGREES	Converts angle in Radians to Degrees.
DIV	Integer division
EXP	Returns e raised to the power of value mentioned.
FLOOR	Returns the largest integer that is less than or equal to a number
GREATEST	Returns the largest value in the list.
LEAST	Returns the smallest value in the list.
LN	Calculates logarithm to the base e.
LOG	Calculates logarithm to the base e.
LOG10	Calculates logarithm to the base 10.
LOG2	Calculates logarithm to the base 2.
MAX	Returns the largest value in a set.
MIN	Returns the least value in a set.
MOD	Returns the remainder after division of two numbers.
PI	Returns value of **π**
POW	Used for exponents.
POWER	Used for exponents.
RADIANS	Converts angle in Degree to Radians.
RAND	Generates a random number.
ROUND	Rounds the number to the nearst decimal place.
SIGN	Returns the sign of a number
SIN	Returns the sine.
SQRT	Returns the root of a number.
SUM	Calculates the sum of a set.
TAN	Returns the tangent.

- **MYSQL DATE FUNCTION**

Function	Description
ADDDATE	Adds a date interval and return the value.
ADDTIME	Adds a time interval and then returns the value.
CURDATE	Returns today’s date
CURRENT_DATE	Same as CURDATE
CURRENT_TIME	Returns the time at the moment

Function	Description
CURRENT_TIMESTAMP	Returns date and time at the moment.
CURTIME	Returns time at the moment.
DATE	Picks up the date from an expression of Date/Time.
DATEDIFF	Returns number of days between two given dates.
DATE_ADD	Similar to ADDDATE
DATE_FORMAT	Changes the format in which Date is displayed.
DATE_SUB	Subtracts a time interval and returns the value.
DAY	Returns the weekday for today.
DAYNAME	Returns the weekday name for any date.
DAYOFMONTH	Used to retrieve the index of the day of the month of any date.
DAYOFWEEK	Used to retrieve the index of the weekday of any date.
DAYOFYEAR	Used to retrieve the index of the day of a year of any date.
EXTRACT	Extracts a part of any date.
HOUR	Returns the “hours” in a given time.
LAST_DAY	Return the last day of the given month.
LOCALTIME	Returns the date and time at the moment.
LOCALTIMESTAMP	Similar to LOCALTIME.
MAKEDATE	Returns a date based on the year and the no. of days you specify.
MAKETIME	Returns a time based on the hours , minutes and seconds you specify.
MICROSECOND	Returns the microseconds in a given time.
MINUTE	Returns the minutes in a given time.
MONTH	Returns the month on a given date.
MONTHNAME	Same as MONTH but returns the name of the month.
NOW	Returns date and time at the moment.
PERIOD_ADD	Adds a specific number of months.
PERIOD_DIFF	Return the difference between two time periods.
SECOND	Return the seconds in a given time.
SEC_TO_TIME	Returns time in seconds.
STR_TO_DATE	Formats the date based on a particular string.
SUBDATE	Same as DATE_SUB.
SUBTIME	Subtracts a time interval.
SYSDATE	Returns the date/time reflected by the system.
TIME	Returns the time from a date/time value.
TIME_FORMAT	Time is displayed based on a certain format.
TIME_TO_SEC	Returns time in seconds.
TIMEDIFF	Returns the difference between two date-time values.
TO_DAYS	Returns the number of days between amy date and "0000-00-00"

- **ADVANCED MYSQL FUNCTION**

Function	Description
BIN	Returns binary value of a given number.
BINARY	Converts a given string to a binary string.
CAST	Converts data from one data type to another.
COALESCE	Returns the first non-null value in a set or list.
CONV	Converts a number from one number-base system to another
CONVERT	Similar to CAST in working
CURRENT_USER	Returns the user name and host name for the MySQL account that is currently used.
DATABASE	Returns the name of the database currently in use.
IF	IF condition statement.
SESSION_USER	Returns the current MySQL user name and host name.
SYSTEM_USER	Similar to SESSION_USER.
USER	Similar to SESSION_USER.
VERSION	Returns the current version of the MySQL server installed.