MySQL 101: Getting Started

Similar to other programming languages like PHP, JavaScript, HTML, and jQuery, MySQL relies on commenting to execute any commands.

You can write two types of comments in MySQL:

- **Single-Line Comments**: These start with "–". Any text that goes after the dash and till the end of the line will not be taken into account by the compiler.

Example:

```
–Update all:
SELECT * FROM Movies;
```

- **Multi-Line Comments:** These start with /* and end with */. Again, any text that is beyond the slashes lines will be ignored by the compiler.

Example:

```
/*Select all the columns
of all the records
in the Movies table:*/
SELECT * FROM Movies;
```

Keeping this in mind, let's get started with actual coding.

How to Connect to MySQL

To start working with MySQL, you'll need to establish an active SSH session on your server.

```
mysql –u root –p
```

If you didn't set a password for your MySQL root user, you omit the -p switch.

Create a new MySQL User Account

Next, you can create a new test user for practice. To do that, run the following command:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
```

If you need to delete a user later on you, use this command:

```
DROP USER 'someuser'@'localhost';
```

Create a New Database

To set up a new database use this line:

```
CREATE DATABASE yourcoolname
```

You can then view all your databases with this command:

```
mysql> show databases;
```

Later on, you can quickly navigate to a particular database using this command:

```
[root@server ~]# mysql -u root -p mydatabase < radius.sql
```

Delete a MySQL Database

To get rid of a database just type:

```
DROP DATABASE dbName
```

If you are done for the day, just type “exit” in the command line to finish your session.

Essential MySQL Commands

SELECT — choose specific data from your database
UPDATE — update data in your database
DELETE — deletes data from your database
INSERT INTO — inserts new data into a database
CREATE DATABASE — generate a new database
ALTER DATABASE — modify an existing database
CREATE TABLE — create a new table in a database
ALTER TABLE — change the selected table
DROP TABLE — delete a table
CREATE INDEX — create an index (search key for all the info stored)
DROP INDEX — delete an index

Working with Tables

Tables are the key element of MySQL databases as they let you store all the information together in organized rows. Each row consists of columns that feature a specified data type. You have plenty of options for customization using the commands below.

Create a New Simple Table

Use this command to create a new table:

```
CREATE TABLE [IF NOT EXISTS] table_name(
  column_list
);
```

The code snippet below features a table for a list of movies that we want to organize by different attributes:

```
CREATE TABLE movies(
   title VARCHAR(100),
   year VARCHAR(100),
   director VARCHAR(50),
   genre VARCHAR(20),
   rating VARCHAR(100),
);
```

View Tables

Use the next commands to get more information about the tables stored in your database.

show tables — call a list of all tables associated with a database.

DESCRIBE table_name; — see the columns of your table.

DESCRIBE table_name column_name; — review the information of the column in your table.

Delete a Table

To get rid of the table specify the table name in the following command:

```
DROP TABLE tablename;
```

Working With Table Columns

Use columns to store alike information that shares the same attribute (e.g. movie director names). Columns are defined by different storage types:

- CHAR
- VARCHAR
- TEXT

Add New Column

```
ALTER TABLE table
ADD [COLUMN] column_name;
```

Delete/Drop a Column

```
ALTER TABLE table_name
DROP [COLUMN] column_name;
```

Insert New Row

```
INSERT INTO table_name (field1, field2, ...) VALUES (value1,
value2, ...)
```

Select Data from The Row

Specify what kind of information you want to retrieve from a certain row.

```
SELECT value1, value2 FROM field1
```

Add an Additional Selection Clause

Include an additional pointer that indicates what type of data do you need.

```
SELECT * FROM movies WHERE budget='1';
SELECT * FROM movies WHERE year='2020' AND rating='9';
```

Delete a Row

Use SELECT FROM syntax and WHERE clause to specify what rows to delete.

```
DELETE FROM movies WHERE budget='1';
```

Update Rows

Similarly, you can use different clauses to update all or specified rows in your table.

To update all rows:

```
UPDATE table_name
SET column1 = value1,
    ...;
```

To update data only in a specified set of rows you can use WHERE clause:

```
UPDATE table_name
SET column_1 = value_1,
WHERE budget='5'
```

You can also update, select or delete rows using JOIN clause. It comes particularly handy when you need to manipulate data from multiple tables in a single query.

Here's how to update rows with JOIN:

```
UPDATE table_name
INNER JOIN table1 ON table1.column1 = table2.column2
SET column1 = value1,
WHERE budget='5'
```

Edit a Column

You can alter any existing column with the following snippet:

```
ALTER TABLE movies MODIFY COLUMN  number INT(3)
```

Sort Entries in a Column

You can sort the data in all columns and rows the same way you do in Excel e.g. alphabetically or from ascending to descending value.

```
SELECT * FROM users ORDER BY last_name ASC;
SELECT * FROM users ORDER BY last_name DESC;
```

Search Columns

Here's how you can quickly find the information you need using WHERE and LIKE syntax:

```
SELECT * FROM movies WHERE genre LIKE 'com%';
SELECT * FROM movies WHERE title LIKE '%a';
```

You can also exclude certain items from search with NOT LIKE:

```
SELECT * FROM movies WHERE genre NOT LIKE 'hor%';
```

Select a Range

Or you can bring up a certain data range using the next command:

```
SELECT * FROM movies WHERE rating BETWEEN 8 AND 10;
```

Concentrate Columns

You can mash-up two or more columns together with CONCAT function:

```
SELECT CONCAT(first_name, ' ', last_name) AS 'Name', dept FROM
users;
```

Data Types

Data types indicate what type of information you can store in a particular column of your table. MySQL has three main categories of data types:

- Numeric
- Text
- Date/time

Logical Operators

Logical operators enable you to add more than one condition in WHERE clause. This makes them super handy for more advanced search, update, insert and delete queries.

In MySQL you have three main logical operators:

- **AND** — use it to filter records that rely on 1+ condition. This way you can call records that satisfy all the conditions separated by AND.
- **OR** — call records that meet any of the conditions separated by OR.
- **NOT** — review records that do not meet a certain condition (e.g. NOT blue). It's a handy operator from excluding certain data.

Plus, some additional special operators:

- **BETWEEN** — select or search data between a range of set min and max values.
- **LIKE** — compare one record to another. Handy operator for search.
- **IS NULL** — compare some value with a NULL value.
- **IN** — determine if a value or expression matches one of the values on your list.
- **ALL** — compare a value or expression to all other values in a list.
- **ANY** — compare a value or expression to any value in your list according to the specified condition.
- **EXISTS** — test if a certain record exists.

Aggregate Functions

Aggregate functions in MySQL allow you to run a calculation on a set of values and return a single scalar value. In essence, they are a great way to find the needed data faster and organize it better using **GROUP BY** and **HAVING** clauses of the **SELECT** statement.

Below is an overview of these:

MIN

Find the smallest value of the selected column in your table:

```
SELECT MIN (column_name)
FROM table_name
WHERE condition;
```

MAX

Does the opposite and returns the largest value of the selected column:

```
SELECT MAX (column_name)
FROM  table_name
WHERE condition;
```

COUNT

Call up several rows that meet the specified criteria:

```
SELECT COUNT (column_name)
FROM table_name
WHERE condition;
```

AVG

Get the average value of a numeric column that you selected:

```
SELECT AVG (column_name)
FROM table_name
WHERE condition;
```

SUM

Receive a total sum of a numeric column that you selected:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Arithmetic, Bitwise, Comparison, and Compound Operators

Arithmetic Operators	Bitwise Operators	Comparison Operators	Compound Operators
+, -, *, /, %	&, |, ^	= =, <, >, <=, >=, < >	+=, *=, -=, /=, %=, &=, ^-+, | *=