MySQL Commands

Help with SQL commands to interact with a MySQL database

MySQL Locations

- Mac */usr/local/mysql/bin*
- Windows */Program Files/MySQL/MySQL version/bin*
- Xampp */xampp/mysql/bin*

Add mysql to your PATH

```
# Current Session
export PATH=${PATH}:/usr/local/mysql/bin
# Permanantly
echo 'export PATH="/usr/local/mysql/bin:$PATH"' >> ~/.bash_profile
```

On Windows - https://www.qualitestgroup.com/resources/knowledge-center/how-to-guide/add-mysql-path-windows/

String Datatypes

Data Type Syntax	Maximum Size	Explanation
CHAR(size)	Maximum size of 255 characters.	Where **size** is the number of characters to strings. Space padded on right to equal **siz**
VARCHAR(size)	Maximum size of 255 characters. single-line.	Where **size** is the number of characters to string.
TINYTEXT(size)	Maximum size of 255 characters. multi-lines	Where **size** is the number of characters to
TEXT(size)	Maximum size of 65,535 characters. multi-lines	Where **size** is the number of characters to length datatype
MEDIUMTEXT(size)	Maximum size of 16,777,215 characters. multi-lines	Where **size** is the number of characters to

Data Type Syntax	Maximum Size	Explanation
LONGTEXT(size)	Maximum size of 4GB or 4,294,967,295 characters. multi-lines	Where **size** is the number of characters to
BINARY(size)	Maximum size of 255 characters.	Where **size** is the number of binary charac length strings. For example, b'111' and b'1 and 128, respectively
VARBINARY(size)	Maximum size of 255 characters.	Where **size** is the number of characters to string. (Introduced in MySQL 4.1.2)

Numeric Datatypes

Data Type Syntax	Maximum Size	Expla
BIT	Very small integer value that is equivalent to TINYINT(1). Signed values range from -128 to 127. Unsigned values range from 0 to 255.	
TINYINT(m)	Very small integer value. Signed values range from -128 to 127. Unsigned values range from 0 to 255.	
SMALLINT(m)	Small integer value. Signed values range from -32768 to 32767. Unsigned values range from 0 to 65535.	
MEDIUMINT(m)	Medium integer value. Signed values range from -8388608 to 8388607. Unsigned values range from 0 to 16777215.	
INT(m)	Standard integer value. Signed values range from -2147483648 to 2147483647. Unsigned values range from 0 to 4294967295.	
BIGINT(m)	Big integer value. Signed values range from -9223372036854775808 to 1. Unsigned values range from 0 to 18446744073709551615.	
DECIMAL(m,d)	Unpacked fixed point number. m defaults to 10, if not specified. d defaults to 0, if not specified.	Where m is the total number ofdigits afte

Data Type Syntax	Maximum Size	Expla
FLOAT(m,d)	Single precision floating point number.	Where `m` is the total d number ofdigits afte
`DOUBLE(m,d)`	Double precision floating point number.	Where `m` is the total d number ofdigits afte
BOOLEAN	Synonym for `TINYINT(1)`	Treated as a boolean value of 0 is consider any other value iscon

Date/Time Datatypes

Data Type Syntax	Maximum Size	E
DATE	Values range from '1000-01-01' to '9999-12-31'.	Displayed as
DATETIME	Values range from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.	Displayed as HH:MM:SS'.
TIMESTAMP(m)	Values range from '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC.	Displayed as HH:MM:SS'.
`TIME`	Values range from '-838:59:59' to '838:59:59'.	Displayed as
`YEAR[(2|4)]`	Year value as 2 digits or 4 digits.	Default is 4

Large Object (LOB) Datatypes

Data Type Syntax	Maximum Size	Explana
`TINYBLOB`	Maximum size of 255 bytes.	
`BLOB(size)`	Maximum size of 65,535 bytes.	Where size is the numbe store. *(size is optional and was 4.1)*

Data Type Syntax	Maximum Size	Explana
MEDIUMBLOB	Maximum size of 16,777,215 bytes.	
LONGTEXT	Maximum size of 4GB or 4,294,967,295 characters.	

Login

```
mysql -u root -p
```

Show Users

```
SELECT User, Host FROM mysql.user;
```

Create User

```
CREATE USER 'someuser'@'localhost' IDENTIFIED BY 'somepassword';
```

Grant All Priveleges On All Databases

```
GRANT ALL PRIVILEGES ON * . * TO 'someuser'@'localhost';
FLUSH PRIVILEGES;
```

Show Grants

```
SHOW GRANTS FOR 'someuser'@'localhost';
```

Remove Grants

```
REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'someuser'@'localhost';
```

Delete User

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

Exit

```
exit;
```

Show Databases

```
SHOW DATABASES
```

Create Database

```
CREATE DATABASE acme;
```

Delete Database

```
DROP DATABASE acme;
```

Select Database

```
USE acme;
```

Create Table

```
CREATE TABLE users(
id INT AUTO_INCREMENT,
   first_name VARCHAR(100),
   last_name VARCHAR(100),
   email VARCHAR(50),
   password VARCHAR(20),
   location VARCHAR(100),
   dept VARCHAR(100),
   is_admin TINYINT(1),
   register_date DATETIME,
   PRIMARY KEY(id)
);
```

Delete / Drop Table

```
DROP TABLE tablename;
```

Show Tables

```
SHOW TABLES;
```

Insert Row / Record

```
INSERT INTO users (first_name, last_name, email, password, location, dept,
is_admin, register_date) values ('Brad', 'Traversy', 'brad@gmail.com',
'123456','Massachusetts', 'development', 1, now());
```

Insert Multiple Rows

```
INSERT INTO users (first_name, last_name, email, password, location, dept,
is_admin, register_date) values ('Fred', 'Smith', 'fred@gmail.com', '123456', 'New
York', 'design', 0, now()), ('Sara', 'Watson', 'sara@gmail.com', '123456', 'New
York', 'design', 0, now()),('Will', 'Jackson', 'will@yahoo.com', '123456', 'Rhode
Island', 'development', 1, now()),('Paula', 'Johnson', 'paula@yahoo.com',
'123456', 'Massachusetts', 'sales', 0, now()),('Tom', 'Spears', 'tom@yahoo.com',
'123456', 'Massachusetts', 'sales', 0, now());
```

Select

```
SELECT * FROM users;
SELECT first_name, last_name FROM users;
```

Where Clause

```
SELECT * FROM users WHERE location='Massachusetts';
SELECT * FROM users WHERE location='Massachusetts' AND dept='sales';
SELECT * FROM users WHERE is_admin = 1;
SELECT * FROM users WHERE is_admin > 0;
```

Delete Row

```
DELETE FROM users WHERE id = 6;
```

Update Row

```
UPDATE users SET email = 'freddy@gmail.com' WHERE id = 2;
```

Add New Column

```
ALTER TABLE users ADD age VARCHAR(3);
```

Modify Column

```
ALTER TABLE users MODIFY COLUMN age INT(3);
```

Add column after(position)

```
ALTER TABLE users ADD age1 after first_name;
```

Drop Column

```
ALTER TABLE users drop COLUMN age;
```

Order By (Sort)

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

Concatenate Columns

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

Select Distinct Rows

```
SELECT DISTINCT location FROM users;
```

Between (Select Range)

```
SELECT * FROM users WHERE age BETWEEN 20 AND 25;
```

Like (Searching)

```
SELECT * FROM users WHERE dept LIKE 'd%';
SELECT * FROM users WHERE dept LIKE 'dev%';
SELECT * FROM users WHERE dept LIKE '%t';
SELECT * FROM users WHERE dept LIKE '%e%';
```

Not Like

```
SELECT * FROM users WHERE dept NOT LIKE 'd%';
```

IN

```
SELECT * FROM users WHERE dept IN ('design', 'sales');
```

Create & Remove Index

```
CREATE INDEX LIndex On users(location);
DROP INDEX LIndex ON users;
```

New Table With Foreign Key (Posts)

```
CREATE TABLE posts(
id INT AUTO_INCREMENT,
   user_id INT,
   title VARCHAR(100),
   body TEXT,
   publish_date DATETIME DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY(id),
   FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Keys/Constraints

Constraint	Description
NOT NULL	A column can not contain any NULL value
UNIQUE	Does not allow to insert a duplicate value in a column. More than one UNIQUE co a table. Allows multiple NULLs in a column
PRIMARY KEY	Unique data for a specific column. It creates a unique index for accessing the tabl allows for auto-increment.
FOREIGN KEY	It creates a link between two tables by one specific column of both tables. The sp table must be a PRIMARY KEY and referred by the column of another table known
CHECK	It determines whether the value is valid or not from a logical expression.
DEFAULT	A column must contain a value (including a NULL). While inserting data into a tab supplied to a column, then the column gets the value set as DEFAULT.

```
CREATE TABLE table (..., PRIMARY KEY (field1, field2))
CREATE TABLE table (..., FOREIGN KEY (field1, field2) REFERENCES table2
(t2_field1, t2_field2))
ALTER TABLE table ADD PRIMARY KEY (field);
ALTER TABLE table ADD CONSTRAINT constraint_name PRIMARY KEY (field, field2);
```

Add Data to Posts Table

```
INSERT INTO posts(user_id, title, body) VALUES (1, 'Post One', 'This is post
one'),(3, 'Post Two', 'This is post two'),(1, 'Post Three', 'This is post
three'),(2, 'Post Four', 'This is post four'),(5, 'Post Five', 'This is post
five'),(4, 'Post Six', 'This is post six'),(2, 'Post Seven', 'This is post
```

```
seven'),(1, 'Post Eight', 'This is post eight'),(3, 'Post Nine', 'This is post
none'),(4, 'Post Ten', 'This is post ten');
```

INNER JOIN

```
SELECT
  users.first_name,
  users.last_name,
  posts.title,
  posts.publish_date
FROM users
INNER JOIN posts
ON users.id = posts.user_id
ORDER BY posts.title;
```

New Table With 2 Foriegn Keys

```
CREATE TABLE comments(
        id INT AUTO_INCREMENT,
    post_id INT,
    user_id INT,
    body TEXT,
    publish_date DATETIME DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY(id),
    FOREIGN KEY(user_id) references users(id),
    FOREIGN KEY(post_id) references posts(id)
);
```

Add Data to Comments Table

```
INSERT INTO comments(post_id, user_id, body) VALUES (1, 3, 'This is comment
one'),(2, 1, 'This is comment two'),(5, 3, 'This is comment three'),(2, 4, 'This
is comment four'),(1, 2, 'This is comment five'),(3, 1, 'This is comment six'),(3,
2, 'This is comment six'),(5, 4, 'This is comment seven'),(2, 3, 'This is comment
seven');
```

Left Join

```
SELECT
comments.body,
posts.title
FROM comments
LEFT JOIN posts ON posts.id = comments.post_id
ORDER BY posts.title;
```

Join Multiple Tables

```
SELECT
comments.body,
```

```
posts.title,
users.first_name,
users.last_name
FROM comments
INNER JOIN posts on posts.id = comments.post_id
INNER JOIN users on users.id = comments.user_id
ORDER BY posts.title;
```

Aggregate Functions

```
SELECT COUNT(id) FROM users;
SELECT MAX(age) FROM users;
SELECT MIN(age) FROM users;
SELECT SUM(age) FROM users;
SELECT UCASE(first_name), LCASE(last_name) FROM users;
select max(age) from users where age < (select max(age) from users); # For second
highest value
```

Group By

```
SELECT age, COUNT(age) FROM users GROUP BY age;
SELECT age, COUNT(age) FROM users WHERE age > 20 GROUP BY age;
SELECT age, COUNT(age) FROM users GROUP BY age HAVING count(age) >=2;
```