


*Contents*

- 1. Query Editor*
- 2. Setting Up a MySQL User Account*
- 3. Data Types*
- 4. SQL Overview*
- 5. SQL Commands*
- 6. Administrative MySQL Commands*
- 7. Storage Engines*
- 8. SQL Constraints*

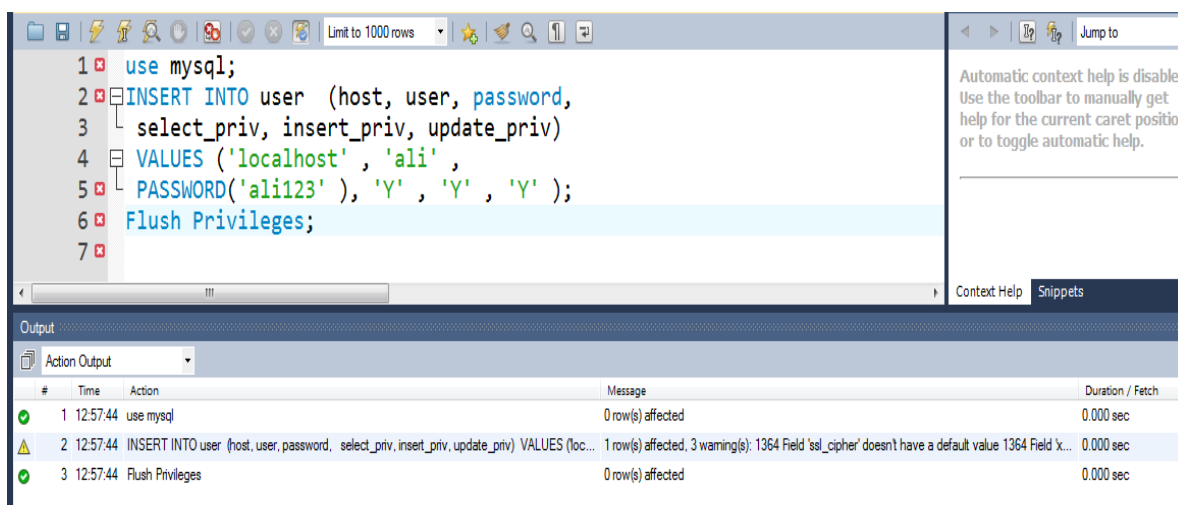
## 1. Query Editor

You can run SQL commands in two different ways:

- CMD: you can run SQL commands by using Command Line Interface (CMD) through **mysql** command.
- MySql Workbench: open *new Query Tab* from menus tab or open  to open query editor.

## 2. Setting up a MySQL User Account

For adding a new user to MySQL, you just need to add a new entry to **user** table in database **mysql**. Below is an example of adding new user **ali** with SELECT, INSERT and UPDATE privileges with the password **ali123**; the SQL query is:



The screenshot shows the MySQL Workbench interface. The top pane is the Query Editor, containing the following SQL script:

```
1 use mysql;
2 INSERT INTO user (host, user, password,
3   select_priv, insert_priv, update_priv)
4 VALUES ('localhost', 'ali',
5   PASSWORD('ali123'), 'Y', 'Y', 'Y');
6 Flush Privileges;
```

The bottom pane is the Output window, showing the execution results:

#	Time	Action	Message	Duration / Fetch
1	12:57:44	use mysql	0 row(s) affected	0.000 sec
2	12:57:44	INSERT INTO user (host, user, password, select_priv, insert_priv, update_priv) VALUES ('loc...	1 row(s) affected, 3 warning(s): 1364 Field 'ssl_cipher' doesn't have a default value 1364 Field 'x...	0.000 sec
3	12:57:44	Flush Privileges	0 row(s) affected	0.000 sec

When adding a new user, remember to encrypt the new password using PASSWORD() function provided by MySQL. As you can see in the above example the password mypass is encrypted to 6f8c114b58f2ce9e.

Notice the `FLUSH PRIVILEGES` statement. This tells the server to reload the grant tables. If you don't use it, then you won't be able to connect to mysql using the new user account at least until the server is rebooted. You can also specify other privileges to a new user by setting the values of following columns in user table to 'Y' when executing the `INSERT` query or you can update them later using `UPDATE` query.

### 3. Data Types

MySQL uses all the standard ANSI SQL numeric data types, so if you're coming to MySQL from a different database system, these definitions will look familiar to you. The following list shows the common numeric data types and their descriptions:

- *Numeric Data Types*
  - **INT**: A normal sized integer that can be signed or unsigned. If signed, the allowable range is from -2147483648 to 2147483647. If unsigned, the allowable range is from 0 to 4294967295. You can specify a width of up to 10 digits (4 bytes).
  - **TINYINT**: A very small integer that can be signed or unsigned. If signed, the allowable range is from -128 to 127. If unsigned, the allowable range is from 0 to 255. You can specify a width of up to 3 digits (1 byte).
  - **SMALLINT**: A small integer that can be signed or unsigned. If signed, the allowable range is from -32768 to 32767. If unsigned, the allowable range is from 0 to 65535. You can specify a width of up to 5 digits (2 bytes).
  - **MEDIUMINT**: A medium-sized integer that can be signed or unsigned. If signed, the allowable range is from -8388608 to 8388607. If unsigned, the

allowable range is from 0 to 16777215. You can specify a width of up to 8 digits (3 bytes).

- **BIGINT**: A large integer that can be signed or unsigned. If signed, the allowable range is from - 9223372036854775808 to 9223372036854775807. If unsigned, the allowable range is from 0 to 18446744073709551615. You can specify a width of up to 20 digits (8 bytes).
- **FLOAT(M,D)**: A floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 10,2, where 2 is the number of decimals and 10 is the total number of digits (including decimals). Decimal precision can go to 24 places (4 bytes) for a FLOAT.
- **DOUBLE(M,D)**: A double precision floating-point number that cannot be unsigned. You can define the display length (M) and the number of decimals (D). This is not required and will default to 16,4, where 4 is the number of decimals. Decimal precision can go to 53 (8 bytes) places for a DOUBLE. **REAL** is a synonym for **DOUBLE**.
- **DECIMAL(M,D)**: An unpacked floating-point number that cannot be unsigned. In unpacked decimals, each decimal corresponds to one byte. Defining the display length (M) and the number of decimals (D) is required.

- *Date and Time Types:*

The MySQL date and time datatypes are:

- **DATE** - A date in YYYY-MM-DD format, between 1000-01 -01 and 9999-12-31. For example, December 30th 1973 would be stored as 1973-12-30.

- DATETIME - A date and time combination in YYYY-MM-DD HH:MM:SS format, between 1000-01 -01 00:00:00 and 9999-12-31 23:59:59. For example, 3:30 in the afternoon on December 30th, 1973 would be stored as 15:30:00 30-12-1973.
- TIMESTAMP: A timestamp between midnight, January 1, 1970 and sometime in 2037. This looks like the previous DATETIME format, only without the hyphens between numbers; 3:30 in the afternoon on December 30th, 1973 would be stored as 19731230153000(YYYYMMDDHHMMSS).
- TIME - Stores the time in HH:MM:SS format.
- YEAR(M) - Stores a year in 2-digit or 4-digit format. If the length is specified as 2 (for example YEAR(2)), (YEAR can be 1970 to 2069 (70 to 69). If the length is specified as 4, YEAR can be 1901 to 2155. The default length is 4.

○ *String Types:*

Although numeric and date types are fun, most data you'll store will be in string format. This list describes the common string datatypes in MySQL.

- CHAR(M) - A fixed-length string between 1 and 255 characters in length (for example CHAR(5)), right padded with spaces to the specified length when stored. Defining a length is not required, but the default is 1.
- VARCHAR(M) - A variable-length string between 1 and 255 characters in length; for example VARCHAR(25). You must define a length when creating a VARCHAR field.
- BLOB or TEXT - A field with a maximum length of 65535 (16 K Bytes) characters. BLOBs are "Binary Large Objects" and are used to store large amounts of binary data, such as images or other types of files. Fields

defined as TEXT also hold large amounts of data; the difference between the two is that sorts and comparisons on stored data are case sensitive on BLOBs and are not case sensitive in TEXT fields. You do not specify a length with BLOB or TEXT.

- **TINYBLOB** or **TINYTEXT**: A BLOB or TEXT column with a maximum length of 255 (8 Bytes) characters. You do not specify a length with TINYBLOB or TINYTEXT.
- **MEDIUMBLOB** or **MEDIUMTEXT**: A BLOB or TEXT column with a maximum length of 16777215(16 M Bytes) characters. You do not specify a length with MEDIUMBLOB or MEDIUMTEXT.
- **LOB** or **LONGTEXT**: A BLOB or TEXT column with a maximum length of 4294967295 (4G Bytes)characters.You do not specify a length with LOB or LONGTEXT.
- **ENUM** - An enumeration, which is a fancy term for list. When defining an ENUM, you are creating a list of items from which the value must be selected (or it can be NULL). For example, if you wanted your field to contain "A" or "B" or "C", you would define your ENUM as ENUM ('A', 'B', 'C') and only those values (or NULL) could ever populate that field. **DECIMAL** is a synonym for **DECIMAL**.

#### 4. SQL Overview

SQL tutorial gives unique learning on **Structured Query Language** and it helps to make practice on SQL commands which provides immediate results. SQL is a language of database, it includes database creation, deletion, fetching rows and

modifying rows etc. SQL is an ANSI (American National Standards Institute) standard, but there are many different versions of the SQL language.

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database. SQL is the standard language for Relation Database System. All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.

Also, they are using different dialects, such as:

- MS SQL Server using T-SQL
- Oracle using PL/SQL
- MS Access version of SQL is called JET SQL (native format) etc.

There are many reasons for using SQL:

- Allows users to access data in relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

## 5. SQL Commands

The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP. These commands can be classified into groups based on their nature:

### 1. DDL - Data Definition Language:

Command	Description
CREATE	Creates a new table, a view of a table, or other object in database
ALTER	Modifies an existing database object, such as a table.
DROP	Deletes an entire table, a view of a table or other object in the database.

### 2. DML - Data Manipulation Language:

Command	Description
INSERT	Creates a record
UPDATE	Modifies records
DELETE	Deletes records

### 3. DCL - Data Control Language:

Command	Description
GRANT	Gives a privilege to user
REVOKE	Takes back privileges granted from user



#### 4. DQL - Data Query Language:

Command	Description
SELECT	Retrieves certain records from one or more tables

#### 6. Administrative MySQL Commands

Here is the list of important MySQL commands, which you will use time to time to work with MySQL database:

- **USE *Databasename*:** This will be used to select a particular database in MySQL work area.
- **SHOW DATABASES:** Lists the databases that are accessible by the MySQL DBMS.
- **SHOW TABLES:** Shows the tables in the database once a database has been selected with the use command.
- **SHOW COLUMNS FROM tablename:** Shows the attributes, types of attributes, key information, whether NULL is permitted, defaults, and other information for a table.
- **SHOW INDEX FROM tablename:** Presents the details of all indexes on the table, including the PRIMARY KEY.
- **SHOW TABLE STATUS [{FROM | IN} db\_name] [LIKE 'pattern' | WHERE expr]** Reports details of the MySQL DBMS performance and statistics.

## 7. Storage Engines

Storage engines are MySQL components that handle the SQL operations for different table types. InnoDB is the default and most general-purpose storage engine, and Oracle recommends using it for tables except for specialized use cases. (The CREATE TABLE statement in MySQL 5.7 creates InnoDB tables by default). MySQL Server uses a pluggable storage engine architecture that enables storage engines to be loaded into and unloaded from a running MySQL server.

To determine which storage engines your server supports, use the statement

```
SHOW ENGINES;
```

The value in the Support column indicates whether an engine can be used. A value of YES, NO, or DEFAULT indicates that an engine is available, not available, or available and currently set as the default storage engine.

- InnoDB: The default storage engine in MySQL 5.7. InnoDB is a **transaction-safe** (ACID (Atomicity, Consistency, Isolation, Durability) compliant) storage engine for MySQL that has **commit, rollback, and crash-recovery capabilities to protect user data**. InnoDB row-level locking (without escalation to coarser granularity locks) and Oracle-style consistent nonlocking reads increase multi-user concurrency and performance. InnoDB stores user data in **clustered indexes** to reduce I/O for common queries based on **primary keys** (With a clustered index the rows are stored physically on the disk in the same order as the index. Therefore, there can be only one clustered index.). To maintain **data integrity**, InnoDB also supports FOREIGN KEY referential-integrity constraints.

- **MyISAM:** These tables have a small footprint. Table-level locking limits the performance in read/write workloads, so it is often used in read-only or read-mostly workloads in Web and data warehousing configurations.
- **Memory:** Stores all data in RAM, for fast access in environments that require quick lookups of non-critical data. This engine was formerly known as the HEAP engine. Its use cases are decreasing; InnoDB with its buffer pool memory area provides a general-purpose and durable way to keep most or all data in memory, and NDBCLUSTER provides fast key-value lookups for huge distributed data sets.
- **CSV:** Its tables are really text files with comma-separated values. CSV tables let you import or dump data in CSV format, to exchange data with scripts and applications that read and write that same format. Because CSV tables are not indexed, you typically keep the data in InnoDB tables during normal operation, and only use CSV tables during the import or export stage.
- **Archive:** These compact, unindexed tables are intended for storing and retrieving large amounts of seldom-referenced historical, archived, or security audit information.
- **Blackhole:** The Blackhole storage engine accepts but does not store data, similar to the Unix /dev/null device. Queries always return an empty set. These tables can be used in replication configurations where DML statements are sent to slave servers, but the master server does not keep its own copy of the data.

- **NDB (also known as NDBCLUSTER):** This clustered database engine is particularly suited for applications that require the highest possible degree of uptime and availability (support generated column constraint).
- **Federated:** Offers the ability to link separate MySQL servers to create one logical database from many physical servers. Very good for distributed or data mart environments.
- **Example:** This engine serves as an example in the MySQL source code that illustrates how to begin writing new storage engines. It is primarily of interest to developers. The storage engine is a “stub” that does nothing. You can create tables with this engine, but no data can be stored in them or retrieved from them.

## 9. SQL Constraints

Constraints are the rules enforced on data columns on table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be column level or table level. Column level constraints are applied only to one column, whereas table level constraints are applied to the whole table.

Following are commonly used constraints available in SQL:

- **NOT NULL Constraint:** Ensures that a column cannot have NULL value.
- **DEFAULT Constraint:** Provides a default value for a column when none is specified.
- **UNIQUE Constraint:** Ensures that all values in a column are different.
- **PRIMARY Key:** Uniquely identified each rows/records in a database table.

- FOREIGN Key: Uniquely identified a rows/records in any another database table.
- CHECK Constraint: The CHECK constraint ensures that all values in a column satisfy certain conditions.
- INDEX: Use to create and retrieve data from the database very quickly.