

MySQL Manual
Lukasz Szybalski

Summary: This paper will cover some strategic features of a MySQL database and its uses. It will show how to calculate mean, standard deviation, how to create a database and a tables, as well as how to get data in and out.

To do any kind of work with a database we need a MySQL database software first, which is free and available to download at www.mysql.com. After downloading it it needs to be configured, accounts need to be created and password protected. To do that we follow instructions given on the mysql website or the installation file that came with a database. After creating and successful log in as "root" you can create a database and then use simple command, written below to create users.

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'test_user'@'localhost'  
-> IDENTIFIED BY 'good_password';
```

This command will create a user with password with an access to a whole database "test". You can specify different privileges for the user, for example privileges without grant option. Also instead of 'localhost' you could specify a domain name or '%' which will allow a user to login from anywhere. After you do that you need to

```
flush privileges;
```

Then log out and log in with the user you just created. You can log in by typing :

```
mysql -u test_user -p
```

This will log in a user called "test_user" and will ask for your password. After successfully logging in, it is required to specify which database you want to use, this can be done by:

```
use database_name;
```

and then follow the procedure that will be mentioned in "1" to create tables. After creating such table you need to load in the file and this can be done by **LOAD DATA INFILE** command. There is a security risk that is associated with that, and therefore, on a newer versions of MySQL this feature is disabled for users. Therefore, it must be necessary to login as "root" to read in the file or ask the administrator to enable it for users. The file that is being loaded in from should be located at the same place where the database is located. If on the other hand, command **LOAD DATA LOCAL INFILE** is enabled then it is possible to specify the directory where the data is and read it in from that folder.

After loading in the data it is straight forward to play with a data that is in a table. You can do a variety of statements. Some of them listed below. The syntax for a simple SQL statement starts out with a selection of things you want to include:

```
Select *  
Select speed, volume
```

after that the statement needs to know where the data should be taken from:

```
from events
```

and at the end you place other restrictions that are needed for the data:

```
group by speed;  
group by distinct speed;  
group by speed where speed < 60;
```

At the end you get a statement that is written as :

```
Select * from events group by speed where speed < 60;
```

Each SQL statement ends with a ";" There is a possibility to limit the amount of what is being printed by adding "LIMIT 10;" at the end of statement. MySQL has few very useful functions that might be used.

These functions are *avg()*, *std()*, *sum()* etc. You can insert ever variable inside of the parenthesis and it will display what is it that you want an average or standard deviation for. These gets very handy if there is a need for calculating averages in data of large amounts. SQL statements can do that function and then the results can be sent to a different program that will handle further analysis.

MySQL is also capable of handling a variety of formats and a date. If your file contains data in a Unix standard time, then you are able to extract it by using function :

```
select FROM_UNIXTIME(variable_name) from table_name;
```

This will print a data in format like this: *2005-02-10 13:15:00*. Then you are able to extract the data in any kind of way you want by using a function called *Data_format*. In code below it will extract only year.

```
DATE_FORMAT(FROM_UNIXTIME(variable_name),"%Y")
```

MySQL is also able to save any output you generate to a file by using command below:

```
SELECT * INTO OUTFILE 'file_name' FROM tbl_name fields terminated by ',' lines terminated by '\n';
```

This will create the output to the file you specified and will separates the data in a way you tell it to. In this case fields will be separated by comma and lines will be terminated by new line. MySQL is a very nice tool to work with, especially if there is a lot of data that needs to be sorted, analyzed, or divided into a smaller peaces. The commands are very easily remembered and an additional once can be found easily just by going to mysql.com website. Below is a list of three question that can be answered by MySQL.

1 . Here is a sample code how to create a database. Creation is very straight forward. This can be written to a file and passed into MySQL database while logging in:

```
mysql -u user -p < script.sql
```

```
_____script.sql_____
```

```
create database hw;  
use hw;
```

```
create table events(  
fielddeviceid varchar(35) Not null,  
locationtimestamp int not null,  
devicestatus varchar(30) not null,  
datastatus varchar(30) not null,  
locstatus varchar(30) not null,  
lastupdatetimes bigint not null,  
volumes double not null,  
speeds double not null,  
occupancy double not null,  
id int unsigned Not null auto_increment primary key  
) type=innodb;
```

```
~  
~  
~
```

2. To calculate the mean, standard deviation for the value, speed, occupancy in the table we enter the following code.

```
mysql> select avg(speeds),avg(occupancy),avg(volumes) from events;  
+-----+-----+-----+  
| avg(speeds)      | avg(occupancy) | avg(volumes)    |  
+-----+-----+-----+  
| 19.045107081275 | 6.8951660170301 | 210.84243482613 |  
+-----+-----+-----+  
1 row in set (0.96 sec)
```

The mean for the speed is 19.04, mean for the occupancy is 6.895, and the mean for volumes is 210.8

```
mysql> select std(speeds),std(occupancy),std(volumes) from events;
+-----+-----+-----+
| std(speeds) | std(occupancy) | std(volumes) |
+-----+-----+-----+
| 11.572388643731 | 8.9949443998595 | 180.83055027602 |
+-----+-----+-----+
1 row in set (0.73 sec)
```

The Standard deviation for speed is 11.572, the standard deviation for occupancy is 8.99, and the standard deviation for volumes is 180.8. The standard deviation for volumes is very big compared to speed and occupancy.

3. Mean and standard deviation for volumes, speed, occupancy for each distinct sensor can be done in a following way. The averages for speed, occupancy, and volume are different for each sensors but are similar in values. There are few missing values for the sensors. Furthermore, there are 736 distinct sensors and on average 222 observations in each. There are 7 sensors which have more then 2,000 observations each. Here are first 10 sensors and their average speed, occupancy, and volumes.

```
mysql> select avg(speeds),avg(occupancy),avg(volumes) from events group by fielddeviceid limit 10;
+-----+-----+-----+
| avg(speeds) | avg(occupancy) | avg(volumes) |
+-----+-----+-----+
|          0 |          0 |          0 |
|          0 |          0 |          0 |
|          0 |          0 |          0 |
| 21.18717970045 | 7.5883299315315 | 224.64864864865 |
| 19.739397925676 | 7.8039012527027 | 214.85135135135 |
|          0 |          0 |          0 |
| 20.82982477027 | 8.7197726432432 | 240.87837837838 |
| 23.271661126126 | 8.0311505837838 | 264.28378378378 |
| 24.858908477477 | 5.6346078621622 | 197.24324324324 |
| 21.732216957207 | 7.9503141540541 | 240.85135135135 |
+-----+-----+-----+
10 rows in set (0.94 sec)
```

Here is how to get a sample standard deviation for first 10 sensors. As you can see some sensors have missing values, while others seem to be constant in their averages. Speed standard deviation is between 2 and 4, standard deviation of occupancy is between 2 and 5 while on average 2, and standard deviation for volume is between 89 and 100.

```
mysql> select std(speeds),std(occupancy),std(volumes) from events group by fielddeviceid limit 10;
+-----+-----+-----+
| std(speeds) | std(occupancy) | std(volumes) |
+-----+-----+-----+
|          0 |          0 |          0 |
|          0 |          0 |          0 |
|          0 |          0 |          0 |
| 3.4128545330504 | 2.7096310555454 | 89.111141116848 |
| 3.0223786729959 | 2.8761190551127 | 85.972132887983 |
|          0 |          0 |          0 |
| 3.1977438001875 | 5.795723745235 | 91.790736255001 |
| 3.7110107596524 | 2.6070297719114 | 100.90690387951 |
| 2.6105621917151 | 2.2292618634713 | 91.275122787821 |
| 3.5837729004961 | 2.7464377582922 | 90.525744051135 |
+-----+-----+-----+
10 rows in set (1.03 sec)
```

4. To get an hour from Unix standard time and group by hours and give an average for each hour for each distinct sensors we can do the following SQL statement.

```
mysql> select avg(speeds),avg(volumes),avg(occupancy), DATE_FORMAT(FROM_UNIXTIME(lastupdatetimes/1000), "%H")
as hour from events group by fielddeviceid, hour limit 130;
```

```
+-----+-----+-----+-----+
| avg(speeds) | avg(volumes) | avg(occupancy) | hour |
+-----+-----+-----+-----+
| 20.6684805 | 94.2 | 3.68307771 | 00 |
| 16.7266096 | 81.3 | 3.97233829 | 01 |
| 16.410771666 | 88 | 4.2958807333 | 02 |
| 17.4987582 | 99.3 | 4.55705567 | 03 |
| 17.988755454 | 147.818181 | 6.3660215181 | 04 |
| 21.4205209 | 323.1 | 10.6403823 | 05 |
| 24.5165061 | 363.6 | 10.1703705 | 06 |
| 23.6805887 | 360.3 | 10.4560726 | 07 |
| 21.4132147 | 341.4 | 11.0866082 | 08 |
| 19.2455349 | 288.6 | 10.5325442 | 09 |
| 18.5989471 | 293.1 | 11.1710429 | 10 |
| 18.4310705 | 243 | 9.3015375 | 11 |
| 19.117867 | 256.2 | 9.60270286 | 12 |
| 18.683570421 | 258.8571428 | 10.031719 | 13 |
| 19.68834935 | 270.6 | 9.87296661 | 14 |
| 22.474884454 | 261.2727272 | 8.2558722636 | 15 |
| 25.2010308 | 247.5 | 6.84794941 | 16 |
| 23.6959612 | 239.7 | 7.27400725 | 17 |
| 24.2160569 | 240.9 | 7.06507874 | 18 |
| 23.0436698 | 202.5 | 6.44166266 | 19 |
| 23.015384 | 183.9 | 5.89612909 | 20 |
| 23.0967843 | 206.4 | 6.51260414 | 21 |
| 21.5410433 | 181.2 | 6.209073 | 22 |
| 22.4078352 | 108.9 | 3.72648738 | 23 |
```

And the standard deviation for the same cases are :

```
mysql> select std(speeds),std(volumes),std(occupancy), DATE_FORMAT(FROM_UNIXTIME(lastupdatetimes/1000), "%H") as
hour from mcs group by fielddeviceid, hour limit 130;
```

```
+-----+-----+-----+-----+
| std(speeds) | std(volumes) | std(occupancy) | hour |
+-----+-----+-----+-----+
| 3.7643679143755 | 13.295111883696 | 0.78597023959676 | 00 |
| 2.5692445338055 | 9.716480844421 | 0.5387582133164 | 01 |
| 1.4381738592137 | 9.8994949366117 | 0.68093703704458 | 02 |
| 3.627734848412 | 10.77079384261 | 0.66860505436237 | 03 |
| 2.0608699042738 | 40.99627074352 | 1.8761273186179 | 04 |
| 3.0700441154914 | 34.048347977545 | 1.2739862207575 | 05 |
| 1.4820417537616 | 22.481992794234 | 0.77830248246837 | 06 |
| 1.5804736964046 | 28.663740160698 | 0.9244647565715 | 07 |
| 1.9650212199595 | 28.548905408089 | 0.99195928255606 | 08 |
| 1.1769177351163 | 15.402597183592 | 0.76490198256936 | 09 |
| 2.2281046331407 | 23.123364807052 | 1.2916183767215 | 10 |
| 0.90210149999999 | 15 | 1.0862495 | 11 |
| 1.8065291830216 | 38.212039987418 | 1.4697641692875 | 12 |
| 1.878118327909 | 20.890384469064 | 1.2314294546069 | 13 |
| 2.6355541956552 | 26.352229507197 | 2.013540060239 | 14 |
| 1.5828033328171 | 19.164792099012 | 1.0803613586004 | 15 |
| 1.6811977384593 | 18.698930450697 | 0.44103600893456 | 16 |
| 1.3672956264077 | 18.122085972647 | 0.50762016347627 | 17 |
| 2.0308915384845 | 18.495675170158 | 1.0008036286703 | 18 |
| 3.3814678889976 | 22.817756243768 | 0.85508630186394 | 19 |
| 1.9049630374565 | 21.215324649885 | 0.81878062708806 | 20 |
| 1.8455266247749 | 21.41588195709 | 0.87829019283256 | 21 |
| 1.545128940847 | 18.792551716039 | 0.52918111659551 | 22 |
| 2.9021708358245 | 19.992248497855 | 0.59854834532409 | 23 |
```

Using MySQL is very trivial. It is very fast when dealing with a lot of data, and the SQL code is easily to remember without much trouble. In conclusion, the tool is very usefully. It is able to do things a normal office software would take hours to do, not to mention any manipulation of a data that you are able to do just by giving a simple command. MySQL is a free, easy, and fast database that can save one a lot of work.