My SQL Function

Objectives

1. Mathematical functions
2. Aggregate functions
3. String functions
4. Date and time functions
5. System functions

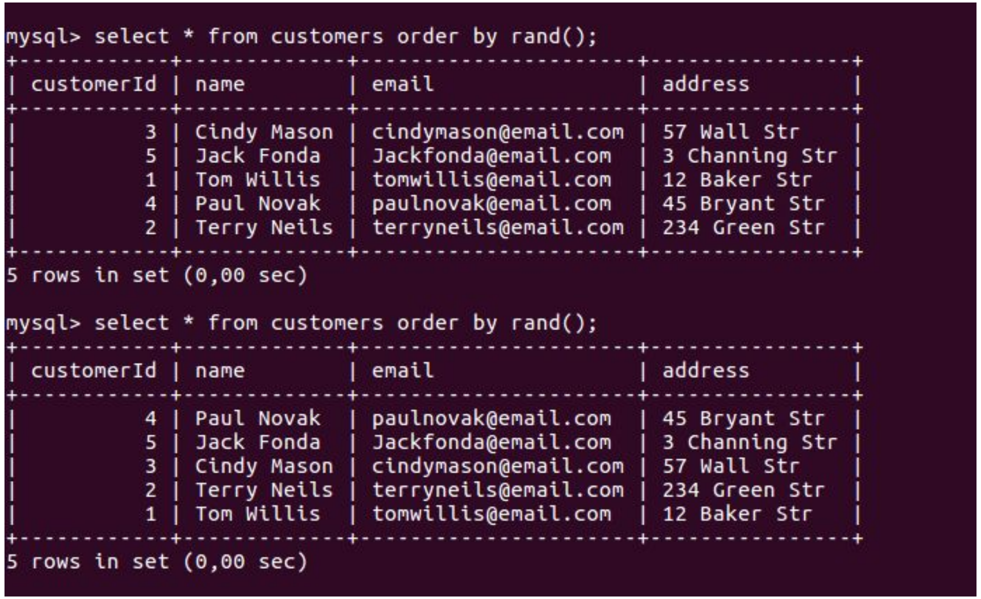
1. Mathematical functions

MySQL supports multiple mathematical functions.

The `RAND()` function returns a **random** number from the <0, 1> interval.

```
mysql> SELECT RAND();
+-------------------+
| RAND()            |
+-------------------+
| 0.786536605829873 |
+-------------------+
```

Or use `ORDER BY RAND()` to get random result:



```
mysql> select * from customers order by rand();
+------------+-------------+----------------------+-----------------+
| customerId | name        | email                | address         |
+------------+-------------+----------------------+-----------------+
|          3 | Cindy Mason | cindymason@email.com | 57 Wall Str     |
|          5 | Jack Fonda  | Jackfonda@email.com  | 3 Channing Str  |
|          1 | Tom Willis  | tomwillis@email.com  | 12 Baker Str    |
|          4 | Paul Novak  | paulnovak@email.com  | 45 Bryant Str   |
|          2 | Terry Neils | terryneils@email.com | 234 Green Str   |
+------------+-------------+----------------------+-----------------+
5 rows in set (0,00 sec)

mysql> select * from customers order by rand();
+------------+-------------+----------------------+-----------------+
| customerId | name        | email                | address         |
+------------+-------------+----------------------+-----------------+
|          4 | Paul Novak  | paulnovak@email.com  | 45 Bryant Str   |
|          5 | Jack Fonda  | Jackfonda@email.com  | 3 Channing Str  |
|          3 | Cindy Mason | cindymason@email.com | 57 Wall Str     |
|          2 | Terry Neils | terryneils@email.com | 234 Green Str   |
|          1 | Tom Willis  | tomwillis@email.com  | 12 Baker Str    |
+------------+-------------+----------------------+-----------------+
5 rows in set (0,00 sec)
```

1. Mathematical functions

The `ABS([number])` function returns the absolute value of a number.

The `PI()` function gives the value of 'π'.

And the `SIN([number])` function computes the sine of an argument.

```
mysql> SELECT ABS(-3), PI(), SIN(0.5);
+---------+----------+-------------------+
| ABS(-3) | PI()     | SIN(0.5)          |
+---------+----------+-------------------+
|       3 | 3.141593 | 0.479425538604203 |
+---------+----------+-------------------+
```

SQL also have functions to give binary, octal and hexadecimal representation of decimal 22.

```
mysql> SELECT BIN(22), OCT(22), HEX(22);
+---------+---------+---------+
| BIN(22) | OCT(22) | HEX(22) |
+---------+---------+---------+
| 10110   | 26      | 16      |
+---------+---------+---------+
```

1. Mathematical functions

The `CEIL([number])` function rounds the value to the smallest following integer.
The `FLOOR([number])` function rounds the value to the largest previous integer.
The `ROUND([number])` returns a number rounded to a specified number of decimal places.

```
mysql> SELECT CEIL(11.256), FLOOR(11.256), ROUND(11.256, 2);
+--------------+---------------+------------------+
| CEIL(11.256) | FLOOR(11.256) | ROUND(11.256, 2) |
+--------------+---------------+------------------+
|           12 |            11 |            11.26 |
+--------------+---------------+------------------+
```

The power and the square root functions.

```
mysql> SELECT POW(3, 3), SQRT(9);
+-----------+---------+
| POW(3, 3) | SQRT(9) |
+-----------+---------+
|        27 |       3 |
+-----------+---------+
```

The `DEGREES([number])` function computes degrees from radians.

```
mysql> SELECT DEGREES(PI());
+---------------+
| DEGREES(PI()) |
+---------------+
|           180 |
+---------------+
1 row in set (0,00 sec)
```

2. Aggregate functions

Aggregate functions operate on sets of values. It useful when statistic data.

First, we have the Cars table.

```
mysql> SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

Want to find out which car is the most expensive and the cheapest, also the average price of these cars?

Use the `MIN([column name])`, `MAX([column name])` and `AVG([column name])` aggregate functions to compute the minimal price, maximal price and the average price of cars in the table.

```
mysql> SELECT MIN(Cost), MAX(Cost), AVG(Cost) FROM Cars;
+-----------+-----------+------------+
| MIN(Cost) | MAX(Cost) | AVG(Cost)  |
+-----------+-----------+------------+
|      9000 |    350000 | 72721.1250 |
+-----------+-----------+------------+
```

2. Aggregate functions

There are some more math function:

```
mysql> SELECT SUM(Cost), COUNT(Id), STD(Cost),
          FROM Cars;
+-----------+----------+------------+
| SUM(Cost) | COUNT(Id) | STD(Cost)   |
+-----------+----------+------------+
|    581769 |         8 | 105931.1676 |
+-----------+----------+------------+
```

`SUM([column name])` function to get the sum of all values in the data set.

`COUNT([column name])` function will count the number of data's row.

`TRUNCATE([number], [decimal places])` function use to return a [number] truncated to a certain number of [decimal] number of decimal places.

...

3. String functions

In this group we have various strings related functions.

```
mysql> SELECT LENGTH('ZetCode'), UPPER('ZetCode'), LOWER('ZetCode');
+-------------------+------------------+-------------------+
| LENGTH('ZetCode') | UPPER('ZetCode') | LOWER('ZetCode')  |
+-------------------+------------------+-------------------+
|                 7 | ZETCODE          | zetcode           |
+-------------------+------------------+-------------------+
```

`LENGTH([string])` function returns the length of a string.

`UPPER([string])` function converts characters into upper-case letters.

`LOWER([string])` function converts characters into lower-case letters.

3. String functions

LPAD([string], [length], [pad string]): Return string that append and prepend characters in [pad string] to the left of [string] with [length]
RPAD() is the same but to the right.

The "sql learning" string has 12 characters.

The LPAD() function appends 8 ' * ' characters to the string, which will be now 20 characters long.

```
mysql> select lpad('sql learning', 20, '*');
+-------------------------------+
| lpad('sql learning', 20, '*') |
+-------------------------------+
| ********sql learning          |
+-------------------------------+
1 row in set (0,00 sec)
```

Or RPAD()

```
mysql> select rpad('sql learning', 20, 'abcd');
+----------------------------------+
| rpad('sql learning', 20, 'abcd') |
+----------------------------------+
| sql learningabcdabcd             |
+----------------------------------+
1 row in set (0,00 sec)
```

3. String functions

The `REVERSE()` function reverses the characters in a string.

The `REPEAT()` function repeats a string specified number of times.

```
mysql> SELECT REVERSE('ZetCode'), REPEAT('*', 6);
+--------------------+----------------+
| REVERSE('ZetCode') | REPEAT('*', 6) |
+--------------------+----------------+
| edoCteZ            | ******         |
+--------------------+----------------+
```

The `LEFT([number])` function returns [number] leftmost characters

The `RIGHT([number])` function returns [number] characters from the right.

The `SUBSTRING([string], [start pos], [number of char])` function returns [number of char] characters from the [start pos] position of the [string].

```
SELECT LEFT('ZetCode', 3), RIGHT('ZetCode', 3), SUBSTRING('ZetCode', 3, 3);
+--------------------+---------------------+----------------------------+
| LEFT('ZetCode', 3) | RIGHT('ZetCode', 3) | SUBSTRING('ZetCode', 3, 3) |
+--------------------+---------------------+----------------------------+
| Zet                | ode                 | tCo                        |
+--------------------+---------------------+----------------------------+
```

3. String functions

`STRCMP():` compares two strings and returns 0 if they are the same.

`CONCAT():` concatenates two strings.

```
mysql> SELECT STRCMP('byte', 'byte'), CONCAT('three', ' apples');
+------------------------+---------------------------+
| STRCMP('byte', 'byte') | CONCAT('three', ' apples') |
+------------------------+---------------------------+
|                      0 | three apples              |
+------------------------+---------------------------+
```

`REPLACE([string 1], [string to replace], [replacement string]):` returns a string, in which we have replaced some text.

[string to replace] is the original string.

[string to replace] is string that we want to replace.

[replacement string] is the new replacing string.

```
mysql> SELECT REPLACE('basketball', 'basket', 'foot');
+-----------------------------------------+
| REPLACE('basketball', 'basket', 'foot') |
+-----------------------------------------+
| football                                |
+-----------------------------------------+
```

4. Date & Time functions

In this group we have various date and time functions.

```
SELECT DAYNAME('2011-01-23'), YEAR('2011/01/23'), MONTHNAME('110123');
+----------------------+--------------------+---------------------+
| DAYNAME('2011-01-23') | YEAR('2011/01/23') | MONTHNAME('110123') |
+----------------------+--------------------+---------------------+
| Sunday               |               2011 | January             |
+----------------------+--------------------+---------------------+
```

In MySQL, date is written in the format YYYY-MM-DD.

Year is followed by month and day. They can be separated by slash or by hyphen.

MySQL also supports a shortened date format, without separators.

Time is written in a standard form, HH:MM:SS. Hours followed by minutes and seconds.

4. Date & Time functions

NOW() function: returns the current date and time.

CURTIME() function: returns the current time.

CURDATE() returns the current date.

```
mysql> select NOW(), CURTIME(), CURDATE();
+---------------------+-----------+------------+
| NOW()               | CURTIME() | CURDATE()  |
+---------------------+-----------+------------+
| 2018-06-05 15:29:41 | 15:29:41  | 2018-06-05 |
+---------------------+-----------+------------+
1 row in set (0,00 sec)
```

4. Date & Time functions

With the `DATEDIFF()` we get the number of days between two dates.

The `DAYNAME()` function returns the day name of a date.

The `MONTHNAME()` function returns a month name of a date.

```
mysql> SELECT DATEDIFF('2011-3-12', '2011-1-12'), DAYNAME('1982-4-12'), MONTHNAME('1982-4-12') ;
+------------------------------------+---------------------+-----------------------+
| DATEDIFF('2011-3-12', '2011-1-12') | DAYNAME('1982-4-12') | MONTHNAME('1982-4-12') |
+------------------------------------+---------------------+-----------------------+
|                                 59 | Monday               | April                  |
+------------------------------------+---------------------+-----------------------+
1 row in set (0,00 sec)
```

4. Date & Time functions

January 23, 2011 can be written in a shortened date format, 110123.
We use the WEEKOFYEAR([date]) to find out the week of the year.
The WEEKDAY([date]) returns 6, which is Sunday.
And the QUARTER([date]) function returns the quarter of the year.
To display date in a different format, we use the DATE_FORMAT([date], [format]).

```
mysql> SELECT WEEKOFYEAR('110123'), WEEKDAY('110123'), QUARTER('110123'), DATE_FORMAT('110123', '%d-%m-%Y');
+----------------------+-------------------+-------------------+-----------------------------------+
| WEEKOFYEAR('110123') | WEEKDAY('110123') | QUARTER('110123') | DATE_FORMAT('110123', '%d-%m-%Y') |
+----------------------+-------------------+-------------------+-----------------------------------+
|                    3 |                 6 |                 1 | 23-01-2011                        |
+----------------------+-------------------+-------------------+-----------------------------------+
```

In DATE_FORMAT([date], [format]), we can also define format by our character:

```
mysql> SELECT DATE_FORMAT('110123', %Y年%m月%d日);
+-----------------------------------+
| DATE_FORMAT('110123', %Y年%m月%d日) |
+-----------------------------------+
| 2011年01月23日                     |
+-----------------------------------+
```

4. Date & Time functions

We can use DATE_ADD([date], INTERVAL [value] [unit]) to add time intervals to a date. Use '-' before if you want to minus time instead of add time, or just use DATE_SUB().

SUBDATE([date], INTERVAL [value] [unit]) to subtract time intervals from a date (Like DATE_SUB()).

```
mysql> SELECT DATE_ADD('110123', INTERVAL 45 DAY),  SUBDATE('110309', INTERVAL 45 DAY);
+-------------------------------------+------------------------------------+
| DATE_ADD('110123', INTERVAL 45 DAY) | SUBDATE('110309', INTERVAL 45 DAY) |
+-------------------------------------+------------------------------------+
| 2011-03-09                          | 2011-01-23                         |
+-------------------------------------+------------------------------------+
```

5. System functions

System functions provide some system information about MySQL database.

We get the version of the MySQL database by `VERSION()` function
The `USER()` function returns the user name and the host name provided by the client.

```
mysql> SELECT Version(), User();
+-------------------------+----------------+
| Version()               | User()         |
+-------------------------+----------------+
| 5.7.22-0ubuntu0.16.04.1 | root@localhost |
+-------------------------+----------------+
```

To get the current database name, use `DATABASE()` function:

```
mysql> select database();
+--------------+
| database()   |
+--------------+
| booking_tour |
+--------------+
```

5. System functions

The `CHARSET()` function returns the character set of the argument. The `COLLATION()` returns the collation of the current string argument. They depend on the charset and collation of the client in use.
In this part of the MySQL tutorial, we worked with the built-in MySQL functions.

```
mysql> SELECT CHARSET('ZetCode'), COLLATION('ZetCode');
+--------------------+----------------------+
| CHARSET('ZetCode') | COLLATION('ZetCode') |
+--------------------+----------------------+
| utf8               | utf8_general_ci      |
+--------------------+----------------------+
```

THE
END