Overview

MYSQL provides lot of built in functions which can be used for transformation to meet our requirements.

- SINGLE ROW FUNCTION => It operates on a single row at a time. It returns one result per row.
- MULTIPLE ROW FUNCTION => It operates on groups of rows. It returns one result for a group of rows.

SINGLE ROW FUNCTIONS

A) CHARACTER FUNCTIONS

Allow us to manipulate string values for display. Some of the manipulations are:

1. Concatenating strings,adding text
   mysql> select concat(ename,' is working as ',job) "employee and his job" from emp;
2. Substring from a string(substr/substring)
   It can return specific character or specific length of characters from the specified position. Can take 2 or 3 arguments.
   mysql> select substr('helloworld',5),substr('helloworld',1,1);
3. Position of a string(Instr)
   It may be required to find the position of a character. (it gives only 1st occurrence). Look at the following example.
   mysql> select instr('helloworld','l') as position
4. Length of a String
   mysql> SELECT LENGTH('mary had a little lamb');
5. Left/Right
   Both of these functions we specify string, length of the string to be kept, relative to right or left.
   mysql> select left('hello world',5),right('helloworld',5);

FUNCTIONS	RESULT
Upper(‘ hello world’)	HELLOWORLD
Lower(‘HELLO WORLD’)	helloworld
Substring(‘hello wold’,1,5)	hello
Left(‘hello world’,3)	Hel
right(‘hello world’,3)	rld
Ltrim(‘ data ‘)	data
Rtrim(‘ data ‘)	data
Replace(‘SREE’,’RE’,’WI’)	SWIE
Concat(‘james’,’bond’)	jamesbond
Repeat(‘data’,3)	datadatadata
Lpad(‘data’,10,’*’)	******data
Rpad(‘data’,10,’*’)	data******
Instr(‘mary’,’a’)	2

B) NUMERIC FUNCTIONS

1) MOD

Remainder can be found using the mod function. I want to find an odd empno, the following query can be written.

mysql> select empno from emp where mod(empno,2)=1;

2) SIGN

Returns -1 if negative number, 1 if positive number, 0 if result is zero.
To compare 2 numbers.

mysql> set @v1=10;
mysql> set @v2=20;
mysql> set @v3=20;
mysql> select sign(@v1-@v2),sign(@v2-@v3),sign(@v3-@v1);

3) ROUND/TRUNCATE/CEIL/FLOOR

mysql> select ceil(91.1),floor(91.9),round(100.218,2), truncate(100.218,2);

```
+------------+-------------+-----------------+--------------------+
| ceil(91.1) | floor(91.9) | round(100.218,2) | truncate(100.218,2) |
+------------+-------------+-----------------+--------------------+
| 92 | 91 | 100.22 | 100.21 |
+------------+-------------+-----------------+--------------------+
```

4) ASCII/CHAR/ABS/POWER/SQRT

mysql> select sqrt(625),power(7,3),ascii('a'),char(65 using ascii);

```
+-----------+------------+------------+---------------------+
| sqrt(625) | power(7,3) | ascii('a') | char(65 using ascii) |
+-----------+------------+------------+---------------------+
| 25 | 343 | 97 | A |
+-----------+------------+------------+---------------------+
```

C) DATE FUNCTIONS

FUNCTION	DESCRIPTION
CURDATE() CURRENT_DATE()	SHOWS CURRENT DATE WITHOUT TIME
SYSDATE() NOW()	SHOWS CURRENT DATE WITH TIME
DATEDIFF()	NUMBER OF DAYS BETWEEN 2 DATES
ADDDATE(),DATE_ADD()	ADDS DAYS,MONTHS,YEARS
SUBDATE(),DATE_SUB()	DEDUCTS DAYS,MONTHS,YEARS
LAST_DAY()	LAST DAY OF MONTH SPECIFIED
DAYNAME()	SHOWS DAYS NAME
MONTHNAME	SHOWS MONTH NAME
DATE()	SHOWS ONLY THE DATE PORTION
TIME()	SHOWS ONLY THE TIME PORTION

1) CHANGING DATE VALUES

mysql> ***select date_add(curdate(),interval '1' day) "add 1 day",***
date_add(curdate(),interval '1' month) "add 1 month",
date_add(curdate(),interval '1' year) "add 1 year";

```
+------------+-------------+------------+
| add 1 day | add 1 month | add 1 year |
+------------+-------------+------------+
| 2022-08-13 | 2022-09-12 | 2023-08-12 |
+------------+-------------+------------+
```

```
mysql> select date_sub(curdate(),interval '1' day) "deduct 1 day",
date_sub(curdate(),interval '1' month) "deduct 1 month",
date_sub(curdate(),interval '1' year) "deduct 1 year";
+--------------+----------------+---------------+
| deduct 1 day | deduct 1 month | deduct 1 year |
+--------------+----------------+---------------+
| 2022-08-11 | 2022-07-12 | 2021-08-12 |
+--------------+----------------+---------------+
mysql> select last_day(curdate());
```

2) Difference between dates

```
mysql> select timestampdiff(month,'2020-08-12',curdate());
+-----------------------------------------+
| timestampdiff(month,'2020-08-12',curdate()) |
+-----------------------------------------+
| 24 |
+-----------------------------------------+
mysql> select timestampdiff(year,'2020-08-12',curdate());
+-----------------------------------------+
| timestampdiff(year,'2020-08-12',curdate()) |
+-----------------------------------------+
| 2 |
+-----------------------------------------+
mysql> select timestampdiff(day,'2020-08-12',curdate());
+-----------------------------------------+
| timestampdiff(day,'2020-08-12',curdate()) |
+-----------------------------------------+
| 730 |
+-----------------------------------------+
mysql> SELECT DATEDIFF(curdate(),'2020-08-12');
+---------------------------------+
| DATEDIFF(curdate(),'2020-08-12') |
+---------------------------------+
| 730 |
+---------------------------------+
```

3) Date_Format

FORMAT	DESCRIPTION	EXAMPLE
%b	Number of month	12
%m	Three letter abbreviation	DEC
%M	Month fully spelt out	DECEMBER
%j	Number of days since Jan 1	347
%d	Number of the day of month	13
%D	DAY NUMBER WITH th,rd etc	13TH
%a	Three-letter abbreviation of day	WED
%W	Day fully spelt out	WEDNESDAY
%Y	4 digit year	1995
%y	2 digit year	95

D) CONTROL FUNCTIONS

1) CASE

There are two versions of using CASE statement:

Version 1:
CASE value WHEN [compare_value]
THEN result [WHEN [compare_value]
THEN result ...] [ELSE result] END
It has been decided to pay bonuses based upon jobs.
For clerk job 1.5 times the salary, analyst 1.75 times the salary, salesman 2.0 times the salary and others only salary as bonus. Let us use CASE to arrive at the bonus.

```
mysql> select ename,job,sal,
-> (case when job='clerk' then 1.5*sal
-> when job='analyst' then 1.75*sal
-> when job='salesman' then 2.0*sal
-> else
-> sal
-> end) "bonus"
-> from emp
-> order by 2;
```

Version 2:
CASE [COL NAME] WHEN [condition] THEN result WHEN [condition] THEN result ...
ELSE result
END

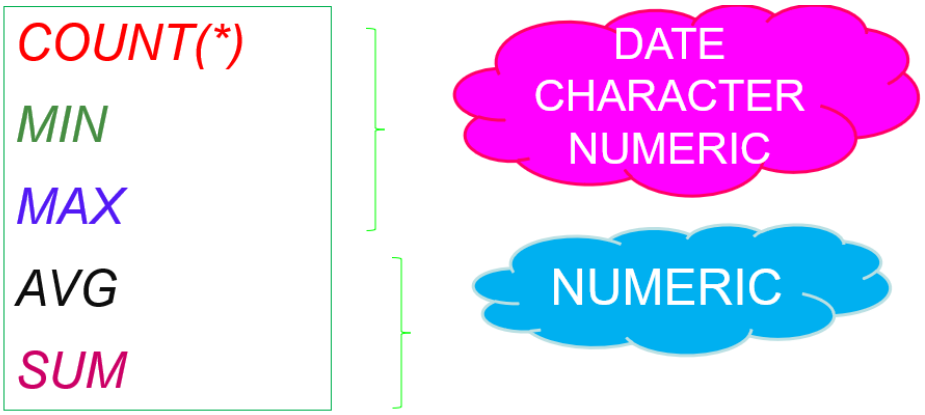
```
mysql> select ename,job,sal,
(case job
when 'clerk' then 1.5*sal
when 'analyst' then 1.75*sal
when 'salesman' then 2.0*sal
else sal
end) "bonus" from emp;
```

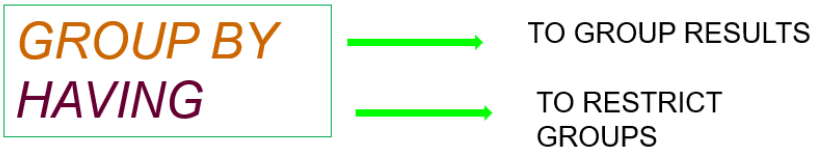
2) **IF(expr1,expr2,expr3)**
IF function accepts three arguments and the result is returned based on if expr1 is TRUE.
If expr1 is evaluated to TRUE, the function returns expr2. Otherwise, expr3 is returned.
mysql> ***select ename,sal,if(sal>3000,'high','low') as comments from emp;***

3) **IFNULL(expr1,expr2)**
If expr1 is not NULL, the function returns expr1. Otherwise it returns expr2.
mysql> ***select ename,sal,comm,sal+comm,ifnull((sal+comm),sal) "ifnull" from emp;***

4) **NULLIF(expr1,expr2)**
Returns NULL if expr1 = expr2 is true, otherwise returns expr1.
mysql> **SELECT ename,sal,length(sal),length(ename), NULLIF(LENGTH(ENAME), LENGTH(SAL)) "nullif" from emp;**

MULTIPLE ROW FUNCTIONS





The GROUP BY clause is used to divide the rows in a table into smaller groups.
The GROUP BY clause is used with the SELECT clause.
SQL groups the result after it retrieves the rows from a table.

Conditional retrieval of rows from a grouped result is possible with the HAVING clause.

The syntax for GROUP BY clause is
SELECT[DISTINCT] <column list> | <expr>
FROM <table>[,<table>] [WHERE condition]
GROUP BY <col | expr>
[HAVING <cond>]

ORDER BY clause can be used to order the final result.

ORDER OF EXECUTION

- It chooses rows based on the where clause
- It groups those rows together based on the group by.
- It calculates the results of the group functions for each group.
- It chooses and eliminates groups based on the having clause
- It orders the groups based on the results of the group functions in the order by. The order must use either a group function or a column in the group by clause.