Table of Contents

1. newsalary_3

For the first two versions of this function the function code just did a simple calculation. Now we are going to add a bit of logic to the function. And we will need to think more about testing our logic.

New rule: Everyone gets the larger of a $3000 or a 10% raise.

1.1. Generating test data to test our function

Let's think about testing our function before we write it. We won't necessarily have the data in the employees table to really test our function. What salary values would we need to use?

If someone has a salary of $10,000, a 10% raise would be 1,000. So this person should get the $3,000 raise- since that is larger.

If someone has a salary of $30,000, a 10% raise would be 3,000. So this person should get the $3,000 raise. Here the 10% and the standard raise of 3000 are the same value.

If someone has a salary of $50,000, a 10% raise would be 5,000. So this person should get the $5,000 raise- since that is larger.

For now we are not going to consider salaries that should be considered invalid such as a zero or negative value- we will get to this.

Test salary value	10% of salary	Standard raise	larger of 10% or standard raise	New salary
10,000	1000	3000	3000	13000
30,000	3000	3000	3000	33000
50,000	5000	3000	5000	55000

Demo 01: Start with a union query. The result is a virtual table with a column name and three rows. The first select in the Union determines the column name.

```
Delimiter #
Select 10000 as tstSalary from dual
union all
Select 30000
union all
Select 50000#
+-----------+
| tstSalary |
+-----------+
|     10000 |
|     30000 |
|     50000 |
+-----------+
```

Demo 02: Use that union query as an inline subquery.

```
Select tstSalary
From (
   select 10000 as tstSalary
   union all
   select 30000
   union all
   select 50000
) as testdata #
```

After we have written our function we can use that subquery as our virtual table.

Everyone gets the larger of a $3000 or a 10% raise- using the MySQL SQL function Greatest. (the Greatest function is not the same as the Max function.)
You can use most of the MySQL intrinsic functions in your MySQL code

Demo 03: newsalary_3

```
Drop function if exists a_emp.newsalary_3#

create function a_emp.newsalary_3 (
   in_salary   decimal (9,2) )
   returns decimal (10,2)
begin
  declare  c_increase_rate       decimal (4,3) default 0.10;
  declare  c_increase_amount     decimal (10,2) default 3000;
  declare  v_using_rate          decimal (10,2);
  declare  v_using_fixed_amount  decimal (10,2);
  declare  v_new_salary          decimal (10,2);

-- what would the value be with a 10% raise
   set v_using_rate:= round(in_salary * (1+ c_increase_rate),2);

-- what would the value be with a $3000 raise
   set v_using_fixed_amount:= in_salary + c_increase_amount;

-- take the bigger of those two values
   set v_new_salary := greatest(v_using_rate, v_using_fixed_amount);

   return v_new_salary ;
end;
#
```

Demo 04: Using the function; this is not the best way to test your function since you are not controlling the values for salary..

```
Select emp_id, salary, dept_id, hire_date
, a_emp.newsalary_3(salary) Sal_3
From a_emp.employees
Order by salary#
+--------+----------+---------+------------+----------+
| emp_id | salary   | dept_id | hire_date  | Sal_3    |
+--------+----------+---------+------------+----------+
|    150 |  6500.00 |      80 | 2001-10-28 |  9500.00 |
|    103 |  9000.00 |     210 | 2010-08-01 | 12000.00 |
|    108 | 12000.00 |      30 | 1995-04-14 | 15000.00 |
|    161 | 15000.00 |     215 | 2011-06-15 | 18000.00 |
|    160 | 15000.00 |     215 | 2011-06-15 | 18000.00 |
```

Demo 05: Testing the function against our generated test data

```
Select tstSalary , a_emp.newsalary_3(tstSalary) as "new salary"
From (
      select 10000 as tstSalary
   union all
      select 30000
   union all
      select 30345.16
   union all
      select 50000) as testdata #
+-----------+------------+
| tstSalary | new salary |
+-----------+------------+
|  10000.00 |   13000.00 |
|  30000.00 |   33000.00 |
|  30345.16 |   33379.68 |
|  50000.00 |   55000.00 |
+-----------+------------+
```

We could add more rows for testing; I added a test row that is not integral. The test table is a virtual table. We are not adding additional tables to the database.

1.2. Improving the testing

When you think about testing, you should decide on what the anticipated return value of the function should be. These are two more demos that show a better testing query.

Demo 06: Including the anticipated result in the test data

```
Select tstSalary
, AnticipatedValue
, a_emp.newsalary_3(tstSalary) as "new salary"
From (
   Select 10000 as tstSalary
        , 13000 as AnticipatedValue
   union all
   Select 30000, 33000
   union all
   Select 50000, 55000
   ) as testdata #
+-----------+------------------+------------+
| tstSalary | AnticipatedValue | new salary |
+-----------+------------------+------------+
|     10000 |            13000 |   13000.00 |
|     30000 |            33000 |   33000.00 |
|     50000 |            55000 |   55000.00 |
+-----------+------------------+------------+
```

Demo 07: You can add another column to show the difference between the anticipated result and your function result. It is easier to scan that column for any differences.

```
Select
  tstSalary
, AnticipatedValue
, a_emp.newsalary_3(tstSalary) as "new salary"
, AnticipatedValue - a_emp.newsalary_3(tstSalary) as "problem"
from (
   select 10000 as tstSalary, 13000 as AnticipatedValue
```

```
   union all
   select 30000, 33000
   union all
   select 50000, 55000
) as testdata #
+-----------+------------------+------------+---------+
| tstSalary | AnticipatedValue | new salary | problem |
+-----------+------------------+------------+---------+
|     10000 |            13000 |   13000.00 |    0.00 |
|     30000 |            33000 |   33000.00 |    0.00 |
|     50000 |            55000 |   55000.00 |    0.00 |
+-----------+------------------+------------+---------+
```

2. newsalary_4

Demo 08: newsalary_4 Same result using an IF structure . This is a regular block if statement, I'll explain it after the demo series

```
Drop function if exists a_emp.newsalary_4#

create function a_emp.newsalary_4 (
   in_salary  decimal (9,2) )
   returns decimal (10,2)
begin
  declare  c_increase_rate      decimal (4,3) default 0.10;
  declare  c_increase_amount    decimal (10,2) default 3000;
  declare  v_new_salary         decimal (10,2);

   if ( in_salary  * c_increase_rate > c_increase_amount) then
      set v_new_salary := round(in_salary  * (1+ c_increase_rate),2);
   else
      set v_new_salary := in_salary  + c_increase_amount;
   end if;

   return v_new_salary;
end;
#
```

Demo 09: Testing the new version of the function

```
Select
  tstSalary
, AnticipatedValue
, a_emp.newsalary_4(tstSalary) as "new salary"
, AnticipatedValue - a_emp.newsalary_4(tstSalary) as "problem"
From (
   Select 10000 as tstSalary
        , 13000 as AnticipatedValue
   union all
   Select 30000, 33000
   union all
   Select 50000, 55000
) as testdata #
```

The next is a two level subquery in the form clause. It makes the outer most query a bit easier to read, but it is fairly complex and it is easy to get the parentheses wrong with this version.

Demo 10:

```
Select
  tstSalary
, AnticipatedValue
, CalcSalary
, AnticipatedValue - CalcSalary as "problem"
From (
   Select tstSalary
   , AnticipatedValue
   , a_emp.newsalary_4(tstSalary) as CalcSalary
   From (
      Select 10000 as tstSalary, 13000 as AnticipatedValue
      union all
      Select 30000, 33000
      union all
      Select 50000, 55000
      ) as testdata
   ) as Calc  #
```

3. newsalary_5

The increase rate depends on the department based on this chart.

Department	increaseRate
10	2%
30	7.5%
35	7.5%
210	5%
215	5%
Others	2.5%

Before writing the function it is a good idea to set up the virtual test table. Many people will complain at this point that it takes work and time to set up the test table and you just want to write the code. But it is important to do the test code. You will think more clearly about the data before you write the function code. And you always need to test your function.

Demo 11: virtual test table

```
Select
  10000 as Salary
, 10 as dept_id
, 10200 as AnticipatedValue
  union all
Select 10000,  20, 10250
  union all
Select 10000,  30, 10750
  union all
Select 10000,  35, 10750
  union all
Select 10000,  80, 10250
```

```
  union all
 Select 10000, 210, 10500
  union all
 Select 10000, 215, 10500
  union all
 Select 10000,   1, 10250#
+--------+---------+------------------+
| Salary | dept_id | AnticipatedValue |
+--------+---------+------------------+
|  10000 |      10 |            10200 |
|  10000 |      20 |            10250 |
|  10000 |      30 |            10750 |
|  10000 |      35 |            10750 |
|  10000 |      80 |            10250 |
|  10000 |     210 |            10500 |
|  10000 |     215 |            10500 |
|  10000 |       1 |            10250 |
+--------+---------+------------------+
```

Demo 12: newsalary_5 The increase rate depends on the department. We now need two parameters/

```
Drop function if exists a_emp.newsalary_5#

create function a_emp.newsalary_5 (
   in_salary  decimal (9,2)
 , in_dept    int )
   returns decimal (10,2)
begin
  declare  v_increase_rate decimal (4,3);
  declare  v_new_salary    decimal (9,2);

   if (in_dept = 10 )then
     set v_increase_rate:= 0.02;
   elseif (in_dept = 30 or in_dept = 35 )then
      set v_increase_rate:= 0.075;
   elseif (in_dept in (210, 215) ) then
     set v_increase_rate:= 0.05;
   else
      set v_increase_rate:= 0.025;
   end if;

   set v_new_salary := round(in_salary  *(1+ v_increase_rate),2);
   return  v_new_salary;

end;
#
```

Demo 13: Using the function against the employee tables.

```
Select
  emp_id, salary, dept_id, hire_date
, a_emp.newsalary_5(salary, dept_id) Sal_5
From a_emp.employees
Limit 5
#
+--------+----------+---------+------------+-----------+
| emp_id | salary   | dept_id | hire_date  | Sal_5     |
+--------+----------+---------+------------+-----------+
```

```
|    100 | 24000.00 |      10 | 1989-06-17 |  24480.00 |
|    101 | 98005.00 |      30 | 2008-06-17 | 105355.38 |
|    102 | 30300.00 |     215 | 2010-06-12 |  31815.00 |
|    103 |  9000.00 |     210 | 2010-08-01 |   9450.00 |
|    104 | 50000.00 |     210 | 2002-10-25 |  52500.00 |
+--------+----------+---------+------------+-----------+
```

Demo 14: testing with the test virtual table

```
select
  Salary
, dept_id
, AnticipatedValue
, a_emp.newsalary_5(salary, dept_id) as "new salary"
, AnticipatedValue - a_emp.newsalary_5(salary, dept_id) as "problem"
from (
    select 10000 as Salary
        ,     10 as dept_id
        ,  10200 as AnticipatedValue
  union all
    select 10000,  20, 10250
  union all
    select 10000,  30, 10750
  union all
    select 10000,  35, 10750
  union all
    select 10000,  80, 10250
  union all
    select 10000, 210, 10500
  union all
    select 10000, 215, 10500
  union all
    select 10000,   1, 10250
 ) as testdata #
```

3.1. version newsalary_6

The increase rate depends on the department.
but people hired within the current year do not get a raise.

Demo 15: Virtual test table now needs a year hired. For each of the department we now have a test case with a hire date in this year and a hire date in a previous year. Since the function is going to look just at the year, I am not worried about the month and day in the test data.
If you are starting to see that we need a full set of test data, you are correct- we need a full set of test data.

```
    select 10000        as Salary
         ,    10        as dept_id
         , '2011-01-15' as hire_date
         , 10000        as AnticipatedValue
    union all
    select 10000
         ,    10
         , '2010-01-15'
         , 10200
    union all
    select 10000,  20, '2011-01-15', 10000
```

```
   union all
   select 10000,  20, '2010-01-15', 10250
   union all
   select 10000,  30, '2011-01-15', 10000
   union all
   select 10000,  30, '2010-01-15', 10750
   union all
   select 10000,  35, '2011-01-15', 10000
   union all
   select 10000,  35, '2010-01-15', 10750
   union all
   select 10000,  80, '2011-01-15', 10000
   union all
   select 10000,  80, '2010-01-15', 10250
   union all
   select 10000, 210, '2011-01-15', 10000
   union all
   select 10000, 210, '2010-01-15', 10500
   union all
   select 10000, 215, '2011-01-15', 10000
   union all
   select 10000, 215, '2010-01-15', 10500
   union all
   select 10000,   1, '2011-01-15', 10000
   union all
   select 10000,   1, '2010-01-15', 10250
#
```

Demo 16: a_emp.newsalary_6 The increase rate depends on the department.
but people hired within the current year do not get a raise. */

```
Drop function if exists a_emp.newsalary_6#

create function a_emp.newsalary_6 (
   in_salary     decimal (9,2)
 , in_dept      int
 , in_hire_date date )
   returns decimal (10,2)
begin
    declare  v_year_hired  decimal (4,0);
    declare  v_new_salary  decimal (10,2);

    set v_year_hired  := extract(year from in_hire_date);
    set v_new_salary := a_emp.newsalary_5(in_salary, in_dept);
    if v_year_hired = extract(year from curdate())
    then
        set v_new_salary := in_salary;
    end if;
    return  v_new_salary;
end;
#
```

Demo 17: Testing the function against the virtual test data

This code is lengthy- and is in the demo. It follows the same model as in the previous demo agains the virtual table.

Demo 18: Using the function against the employee table.

```
select emp_id, salary, dept_id, hire_date
, a_emp.newsalary_6(salary, dept_id, hire_date)  Sal_6
from a_emp.employees
limit 5
#
+--------+----------+---------+------------+-----------+
| emp_id | salary   | dept_id | hire_date  | Sal_6     |
+--------+----------+---------+------------+-----------+
|    100 | 24000.00 |      10 | 1989-06-17 |  24480.00 |
|    101 | 98005.00 |      30 | 2008-06-17 | 105355.38 |
|    102 | 30300.00 |     215 | 2010-06-12 |  31815.00 |
|    103 |  9000.00 |     210 | 2010-08-01 |   9450.00 |
|    104 | 50000.00 |     210 | 2002-10-25 |  52500.00 |
+--------+----------+---------+------------+-----------+
```

4. Rules for Functions to be Used in SQL statements

- The function should do a straight-forward calculation only.
- The function should have no side effects.
- The function should not do a DML statement such as an Insert or Update. For that you should use a procedure.
- The return type must be a data type allowed in MySQL tables.

5. Logic Control Structures : Selection Structures

5.1. Block If

Syntax model for the block if structure.

```
IF  logical_test1 THEN
   True_path1_statements
ELSEIF logical_test2 THEN
   True_path2_statements
ELSEIF logical_test3 THEN
   True_path3_statements
/* you can have as many ELSEIF blocks as needed */
ELSE
   Not_true_path_statements
END IF;
```

The block if uses the key word If to start the structure and the key word End If to end it. After the word IF you can have a logical expression – such as those you would use in a row filter; this can be a simple or complex expression. Then we have the word THEN. This is followed by one or more statements that should be executed if that logical expression has the value True.

If the first logical expression is not True, then the second logical expression is evaluated.

If none of the logical expressions evaluate to True, then the statements after the key word Else are executed.

A few things to note

- If a logical expression evaluates to unknown because of a null, then that expression is not treated as true when the test expression is evaluated.
- The Else block is not required
- The first logical expression that evaluates to true wins; its code is executed and the program flow transfers to the statement after End if
- There are no semi-colons after the word Then; There is a semi-colon after the End IF

5.2. Case structure

MySQL includes a Case statement. You can also use the simple Case and the searched Case expressions discussed in the SQL section.

Syntax model for a simple case statement to perform an action. Note that the ending phrase is END CASE.

```
--here we are assuming that the variables v_counter   and v_varb have been
declared and that v_counter has a value.
-- You can have as many When … then block as you need
      CASE  v_counter
         WHEN 1 THEN
            Set v_varb  := 'ONE';
         WHEN 2 THEN
            Set v_varb  := 'TWO';
         ELSE
            Set v_varb := 'MANY';
      END CASE;

END;
```

With the case statement, if there is no Else clause and none of the case tests are matched, an error occurs (`ERROR 1339 (20000): Case not found for CASE statement`)

Demo 19:

```
Drop function if exists a_emp.newsalary_5_V2#
create function a_emp.newsalary_5_V2 (
   in_salary  decimal (9,2)
 , in_dept    int )
   returns decimal (10,2)
begin
  declare  v_increase_rate decimal (4,3);
  declare  v_new_salary    decimal (9,2);

  case in_dept
    when 10 then
       set v_increase_rate:= 0.02;
    when 30 then
        set v_increase_rate:= 0.075;
    when 35 then
       set v_increase_rate:= 0.075;
    when 210 then
       set v_increase_rate:= 0.05;
    when 215 then
       set v_increase_rate:= 0.05;
    else
        set v_increase_rate:= 0.025;
    end case;

  set v_new_salary := round(in_salary  *(1+ v_increase_rate),2);
  return  v_new_salary;
end;
#
```

Demo 20 is the test for this function. Use the model for testing a_emp.newsalary_5.