Table of Contents

This discusses the following

```
Greatest()                    Least()                       If()
```

1. GREATEST and LEAST

GREATEST and LEAST return the largest and smallest value from the list of arguments. Notice what happens with nulls. If there is a null in the list then the functions treat this as an unknown value and therefore the function cannot "know" which value in the list is the largest.

Demo 01:

```
Select A
,      GREATEST(B,C,D,E,F)
,      LEAST(B, C, D, E,F)
From a_testbed.z_numbers;
+------+---------------------+---------------------+
| A    | GREATEST(B,C,D,E,F) | LEAST(B, C, D, E,F) |
+------+---------------------+---------------------+
|    1 |                  90 |                  10 |
|    2 |                NULL |                NULL |
|    3 |                NULL |                NULL |
|    4 |                NULL |                NULL |
|    5 |                   0 |                   0 |
|    6 |                  10 |                  10 |
|    7 |                 210 |                 -10 |
|    8 |                  85 |                -210 |
|    9 |                 200 |                  -1 |
|   10 |                 200 |                  -1 |
+------+---------------------+---------------------+
```

Data type Issues: A function returns a single value. GREATEST (list), LEAST (list) returns the largest, smallest in the list. Avoid mixing incompatible data types.

Demo 02:

```
Select greatest(4, 45.78, 9);
+------------------------+
| greatest( 4, 45.78, 9) |
+------------------------+
|                  45.78 |
+------------------------+
```

Demo 03:

```
Select greatest( 'p', 4, 45.78, 9);
+-----------------------------+
| greatest( 'p', 4, 45.78, 9) |
+-----------------------------+
| 45.78                       |
+-----------------------------+
1 row in set, 1 warning (0.00 sec)
show warnings;
```

```
+---------+------+---------------------------------------+
| Level   | Code | Message                               |
+---------+------+---------------------------------------+
| Warning | 1292 | Truncated incorrect DOUBLE value: 'p' |
+---------+------+---------------------------------------+
```

MySQL's general approach is to treat type issues as something to be compensated for more than as an error. You need to develop the habit of checking for the warning messages.

Demo 04:

```
Select least( 'p', 4, 45.78, 9);
+--------------------------+
| least( 'p', 4, 45.78, 9) |
+--------------------------+
| 0                        |
+--------------------------+
```

1.1. Examples using the demo tables

Demo 05: Returns items which were sold at a price higher than their list price.

```
Select ord_id
, prod_id
, quoted_price
, prod_list_price
From a_oe.order_details
Join a_prd.products using (prod_id)
Where GREATEST(quoted_price, prod_list_price) > (prod_list_price);
+--------+---------+--------------+-----------------+
| ord_id | prod_id | quoted_price | prod_list_price |
+--------+---------+--------------+-----------------+
|    120 |    1010 |       175.00 |          150.00 |
|    121 |    1010 |       175.00 |          150.00 |
|    390 |    1010 |       175.00 |          150.00 |
|    395 |    1010 |       195.00 |          150.00 |
|    550 |    1010 |       175.00 |          150.00 |
|    551 |    1010 |       175.00 |          150.00 |
|    301 |    1100 |       205.00 |           49.99 |
+--------+---------+--------------+-----------------+
```

2. IF

The IF function takes three arguments; the first should be an expression that has a true/false value. If its value is true then the return value is the value of argument 2; if the value of the first expression is not true then the return value is the value of argument 3.

Demo 06:

```
Select if(curdate() > '1888-08-08', 'passed the test', 'this is really old');
+-----------------------------------------------------------------------+
| if(curdate() > '1888-08-08', 'passed the test', 'this is really old') |
+-----------------------------------------------------------------------+
| passed the test                                                       |
+-----------------------------------------------------------------------+

Select if(month(curdate()) in (6,7,8), 'Summer!', 'Not summer');
+----------------------------------------------------------+
| if(month(curdate()) in (6,7,8), 'Summer!', 'Not summer') |
+----------------------------------------------------------+
| Summer!                                                  |
+----------------------------------------------------------+

Select if( 5 > null, 'passed the test', 'nulls make unknown values');
+--------------------------------------------------------------+
| if(5 > null, 'passed the test', 'nulls make unknown values') |
+--------------------------------------------------------------+
| nulls make unknown values                                    |
+--------------------------------------------------------------+
```

Demo 07: We want to give customers a 5% savings for each pet supply item or sporting goods item. As a first step we will determine the percent to apply to the price.

```
Select catg_id, prod_id, prod_list_price
, if(catg_id IN('PET','SPG'), 0.95, 1) as "Price Multiplier"
From a_prd.products products
Order by prod_id;
```

Selected rows

```
+---------+---------+-----------------+------------------+
| catg_id | prod_id | prod_list_price | Price Multiplier |
+---------+---------+-----------------+------------------+
| HW      |    1000 |          125.00 |                1 |
| SPG     |    1010 |          150.00 |             0.95 |
| SPG     |    1050 |          269.95 |             0.95 |
| SPG     |    1060 |          255.95 |             0.95 |
| APL     |    1125 |          500.00 |                1 |
| APL     |    1130 |          149.99 |                1 |
| PET     |    1140 |           14.99 |             0.95 |
| PET     |    4577 |           29.95 |             0.95 |
+---------+---------+-----------------+------------------+
```

Demo 08: The if test could be nested for another test. Suppose that APL were to get a 10% discount.

```
Select catg_id, prod_id, prod_list_price
, if(catg_id IN('PET','SPG'), 0.95, if(catg_id IN('APL'), 0.90, 1)
) as "Price Multiplier"
From     a_prd.products products
Order by prod_id;
```

Selected rows

```
+---------+---------+-----------------+------------------+
| catg_id | prod_id | prod_list_price | Price Multiplier |
+---------+---------+-----------------+------------------+
| HW      |    1000 |          125.00 |                1 |
| SPG     |    1010 |          150.00 |             0.95 |
| SPG     |    1050 |          269.95 |             0.95 |
| SPG     |    1060 |          255.95 |             0.95 |
| APL     |    1125 |          500.00 |             0.90 |
| APL     |    1130 |          149.99 |             0.90 |
| PET     |    1140 |           14.99 |             0.95 |
| PET     |    4577 |           29.95 |             0.95 |
+---------+---------+-----------------+------------------+
```

Nesting IF tests like this quickly becomes hard to read and is error prone. For anything other than a simple single test. use a case expression. Also case expressions are standard SQL supported by most dbms; the If function is not widely supported.