Table of Contents

The Case expressions are used to perform selection logic. The case expression is part of standard SQL and corresponds closely to selection logic found in most programming languages. The case expression is not a function but it is a bit more complex that the simpler expressions we used in earlier units.

1. Searched Case expression

The searched Case expression requires a logical expression to be evaluated at each WHEN clause.

All of the return expressions must have the same data type or be capable of being cast implicitly to the case of the first argument.

You can use a variety of tests- In lists, Between, wildcard tests and you can mix the tests in a single case expression. You can nest case expressions.

Demo 01: We want to give customers a 5% savings for each pet supply item, 5% for each sporting goods item and 10% for each appliance. As a first step we will determine the percent to apply to the price.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  1
  END  as "Price Multiplier"
from product.products
order by catg_id;
```

selected rows

```
+---------+---------+-----------------+------------------+
| catg_id | prod_id | prod_list_price | Price Multiplier |
+---------+---------+-----------------+------------------+
| APL     |    1120 |          549.99 |             0.90 |
| APL     |    1125 |          500.00 |             0.90 |
| HW      |    1080 |           25.00 |                1 |
| HW      |    1090 |          149.99 |                1 |
| HW      |    1110 |           49.99 |                1 |
| PET     |    1150 |            4.99 |             0.95 |
| PET     |    1152 |           55.00 |             0.95 |
| SPG     |    1010 |          150.00 |             0.95 |
| SPG     |    1030 |           29.95 |             0.95 |
```

Demo 02: We can use that calculated percent to determine the sales price. Note that Case returns a numeric result which we can use in multiplication.

```
select   catg_id, prod_id, prod_list_price
,        CASE
             WHEN catg_id ='PET'   THEN 0.95
             WHEN catg_id ='SPG'   THEN 0.95
             WHEN catg_id ='APL'   THEN 0.90
         ELSE  1
         END  * prod_list_price AS "Today's Price"
from     product.products
order by catg_id;
```

selected rows

```
+---------+---------+-----------------+---------------+
| catg_id | prod_id | prod_list_price | Today's Price |
+---------+---------+-----------------+---------------+
| APL     |    1120 |          549.99 |      494.9910 |
| APL     |    1125 |          500.00 |      450.0000 |
| HW      |    1080 |           25.00 |       25.0000 |
| HW      |    1090 |          149.99 |      149.9900 |
| HW      |    1100 |           49.99 |       49.9900 |
| HW      |    1110 |           49.99 |       49.9900 |
| HW      |    1160 |          149.99 |      149.9900 |
| PET     |    1142 |            2.50 |        2.3750 |
| PET     |    1150 |            4.99 |        4.7405 |
| PET     |    1151 |           14.99 |       14.2405 |
| PET     |    1152 |           55.00 |       52.2500 |
| SPG     |    1010 |          150.00 |      142.5000 |
| SPG     |    1030 |           29.95 |       28.4525 |
| SPG     |    1060 |          255.95 |      243.1525 |
```

Demo 03: You should include an Else clause unless you are certain that all possible values are handled. Here I have removed the else clause and products which do not fall into one of the three categories tested, get a value of null from the case expression and therefore have a null value for the last column. This does not follow the business rule of demo 01

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  END  * prod_list_price AS "Today's Price"
from product.products
order by catg_id
;
+---------+---------+-----------------+---------------+
| catg_id | prod_id | prod_list_price | Today's Price |
+---------+---------+-----------------+---------------+
| APL     |    1120 |          549.99 |      494.9910 |
| APL     |    1130 |          149.99 |      134.9910 |
| APL     |    4569 |          349.95 |      314.9550 |
| APL     |    1125 |          500.00 |      450.0000 |
| APL     |    1126 |          850.00 |      765.0000 |
| GFD     |    5001 |            5.00 |          NULL |
| GFD     |    5000 |           12.50 |          NULL |
| HD      |    5002 |           23.00 |          NULL |
| HD      |    5008 |           12.50 |          NULL |
| HD      |    5004 |           15.00 |          NULL |
```

1.1. Return type consistency

MySQL is a bit more robust than some of the other dbms. Suppose you run the following query; The case expression says that for catg_id of 'PET', 'SPG' and 'APL' we are returning a number and for other categories we are returning a string. In many dbms you would have a problem (an error) since the return type of the expression is not consistent. MySQL continues the query execution and based on the alignment in this client it is returning a string for that column.

Demo 04:

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  'no discount'
  END  "Savings %"
from product.products
order by catg_id
;
+---------+---------+-----------------+-------------+
| catg_id | prod_id | prod_list_price | Savings %   |
+---------+---------+-----------------+-------------+
| APL     |    1120 |          549.99 | 0.90        |
| APL     |    1130 |          149.99 | 0.90        |
| APL     |    4569 |          349.95 | 0.90        |
| APL     |    1125 |          500.00 | 0.90        |
| APL     |    1126 |          850.00 | 0.90        |
| GFD     |    5001 |            5.00 | no discount |
| GFD     |    5000 |           12.50 | no discount |
| HD      |    5002 |           23.00 | no discount |
| HD      |    5008 |           12.50 | no discount |
```

Now go one step further and multiply that case expression by the list price to get Today's Price as we did in a previous query. The result does not show the last column as null (as before) it shows that today all of these items are FREE! (I think you might have just lost your job.)

Demo 05: Note that we do get warnings. This is a warning of an error you need to correct.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  'no discount'
  END  * prod_list_price as "Today's Price"
from product.products
order by catg_id
;
+---------+---------+-----------------+--------------------+
| catg_id | prod_id | prod_list_price | Today's Price      |
+---------+---------+-----------------+--------------------+
| APL     |    1120 |          549.99 | 494.99100000000004 |
| APL     |    1130 |          149.99 |            134.991 |
| APL     |    4569 |          349.95 |            314.955 |
| APL     |    1125 |          500.00 |                450 |
| APL     |    1126 |          850.00 |                765 |
| GFD     |    5001 |            5.00 |                  0 |
| GFD     |    5000 |           12.50 |                  0 |
| HD      |    5002 |           23.00 |                  0 |
Warning (Code 1292): Truncated incorrect DOUBLE value: 'no discount'
Warning (Code 1292): Truncated incorrect DOUBLE value: 'no discount'
```

You could add another column for the no discount message.

```
+---------+---------+-----------------+-----------------+-------------+
| catg_id | prod_id | prod_list_price | Today's Price % |             |
+---------+---------+-----------------+-----------------+-------------+
| APL     |    1120 |          549.99 |          494.99 |             |
| APL     |    1130 |          149.99 |          134.99 |             |
| APL     |    4569 |          349.95 |          314.96 |             |
| APL     |    1125 |          500.00 |          450.00 |             |
| APL     |    1126 |          850.00 |          765.00 |             |
| GFD     |    5001 |            5.00 |            5.00 | no discount |
| GFD     |    5000 |           12.50 |           12.50 | no discount |
| HD      |    5002 |           23.00 |           23.00 | no discount |
| HD      |    5008 |           12.50 |           12.50 | no discount |
| HD      |    5004 |           15.00 |           15.00 | no discount |
| HD      |    5005 |           45.00 |           45.00 | no discount |
| HW      |    1100 |           49.99 |           49.99 | no discount |
| MUS     |    2412 |            9.87 |            9.87 | no discount |
| MUS     |    2746 |           14.50 |           14.50 | no discount |
```

Why did that happen? Because that is the way that MySQL works- every dbms has some oddities. MySQL tries to cast the strings to numbers when it does the multiplication but when it cannot do the cast, it treats the string as a 0 value.

```
select 'abc', 'abc' * 25;
+-----+------------+
| abc | 'abc' * 25 |
+-----+------------+
| abc |          0 |
+-----+------------+
1 row in set, 1 warning (0.00 sec)
Warning (Code 1292): Truncated incorrect DOUBLE value: 'abc'
```

1.2. Including other functions

Demo 06: We can then include the round function to improve the format.

```
select  catg_id, prod_id, prod_list_price
, Round(
   CASE
      WHEN catg_id ='PET'   THEN 0.95
      WHEN catg_id ='SPG'   THEN 0.95
      WHEN catg_id ='APL'   THEN 0.90
   ELSE  1
   END  * prod_list_price, 2 ) AS "Today's Price"
from product.products
order by catg_id;
```

selected rows

```
+---------+---------+-----------------+---------------+
| catg_id | prod_id | prod_list_price | Today's Price |
+---------+---------+-----------------+---------------+
| APL     |    1120 |          549.99 |        494.99 |
| APL     |    1125 |          500.00 |        450.00 |
| HW      |    1080 |           25.00 |         25.00 |
| HW      |    1090 |          149.99 |        149.99 |
| HW      |    1110 |           49.99 |         49.99 |
| PET     |    1142 |            2.50 |          2.38 |
| PET     |    1150 |            4.99 |          4.74 |
| PET     |    1151 |           14.99 |         14.24 |
| PET     |    1152 |           55.00 |         52.25 |
| SPG     |    1010 |          150.00 |        142.50 |
| SPG     |    1030 |           29.95 |         28.45 |
```

In the next example we want the discount to apply only to products with a list price of $50 or higher. The first When clause with a true value determines the result.

Demo 07: The first When clause with a true value determines the result. Items with prices under $50 are not considered for a discount.

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN prod_list_price < 50 THEN 1
     WHEN catg_id ='PET'   THEN 0.95
     WHEN catg_id ='SPG'   THEN 0.95
     WHEN catg_id ='APL'   THEN 0.90
  ELSE  1
  END  * prod_list_price AS "Today's Price"
from product.products
order by catg_id;
```

Selected rows

```
+---------+---------+-----------------+---------------+
| catg_id | prod_id | prod_list_price | Today's Price |
+---------+---------+-----------------+---------------+
| APL     |    1120 |          549.99 |      494.9910 |
| APL     |    1125 |          500.00 |      450.0000 |
| HW      |    1080 |           25.00 |       25.0000 |
| HW      |    1090 |          149.99 |      149.9900 |
| HW      |    1100 |           49.99 |       49.9900 |
| HW      |    1110 |           49.99 |       49.9900 |
| PET     |    1142 |            2.50 |        2.5000 |
| PET     |    1150 |            4.99 |        4.9900 |
| PET     |    1152 |           55.00 |       52.2500 |
| SPG     |    1010 |          150.00 |      142.5000 |
| SPG     |    1030 |           29.95 |       29.9500 |
| SPG     |    1060 |          255.95 |      243.1525 |
```

The next case structure looks daunting in code but look at the output first. With appliances we merely report back that this is an appliance item. With pet supplies and sporting good we break these down into cost categories (high, low, medium). The break points for sporting goods and pet supplies are different. For all other categories we do not report anything.

The outer case structure is based on the category id- there is a block for PET, another block for SPG, a third block for APL and no Else block. Items which do not fit in one of these categories do not get a block and the case returns a null. When you develop this code you should write and test the outer case structure first.

The inner case structure for PET and the inner case structure for SPG are based on the prod_list_price

Demo 08: –A nested Case structure. prd_products

```
select  catg_id, prod_id, prod_list_price
, CASE
     WHEN catg_id ='PET'   THEN
        CASE
           WHEN prod_list_price < 10 THEN 'LowCost pet item'
        ELSE 'HighCost pet item'
        END
     WHEN catg_id ='SPG'   THEN
        CASE
           WHEN prod_list_price < 25 THEN 'LowCost sports item'
           WHEN prod_list_price between 25 and 150 THEN 'MidCost sports item'
        ELSE 'HighCost sports item'
        END
     WHEN catg_id ='APL'   THEN 'appliance item'
  END  AS "Result"
from product.products
order by prod_id;
```

selected rows

```
+---------+---------+-----------------+----------------------+
| catg_id | prod_id | prod_list_price | Result               |
+---------+---------+-----------------+----------------------+
| HW      |    1000 |          125.00 | NULL                 |
| SPG     |    1010 |          150.00 | MidCost sports item  |
| SPG     |    1020 |           12.95 | LowCost sports item  |
| SPG     |    1030 |           29.95 | MidCost sports item  |
| SPG     |    1040 |          349.95 | HighCost sports item |
| HW      |    1090 |          149.99 | NULL                 |
| HW      |    1100 |           49.99 | NULL                 |
| APL     |    1120 |          549.99 | appliance item       |
| APL     |    1130 |          149.99 | appliance item       |
| PET     |    1140 |           14.99 | HighCost pet item    |
| PET     |    1142 |            2.50 | LowCost pet item     |
| PET     |    1150 |            4.99 | LowCost pet item     |
| HW      |    1160 |          149.99 | NULL                 |
| PET     |    4567 |          549.99 | HighCost pet item    |
| PET     |    4568 |          549.99 | HighCost pet item    |
| APL     |    4569 |          349.95 | appliance item       |
| HW      |    4575 |           49.95 | NULL                 |
| PET     |    4577 |           29.95 | HighCost pet item    |
```

If we want to display a message instead of the missing value, we can wrap a coalesce function around the entire case expression.: `Coalesce(CASE  . . . END, 'No information available') as "Result"`

Demo 09: We have a look up table for the credit ratings. This is another approach. If the credit levels for the rating terms were to change frequently, the lookup table would be a better approach.
Note what is returned if the credit_limit is null.

```
select  customer_id, customer_credit_limit
, CASE
     WHEN customer_credit_limit >= 10001  THEN 'Superior'
     WHEN customer_credit_limit >=  5001  THEN 'Excellent'
     WHEN customer_credit_limit >=  2001  THEN 'High'
     WHEN customer_credit_limit >=  1001  THEN 'Good'
  ELSE 'Standard'
  END  AS  Rating
from customer.customers;
```

Selected rows

```
+-------------+-----------------------+-----------+
| customer_id | customer_credit_limit | Rating    |
+-------------+-----------------------+-----------+
|      400300 |                  6000 | Excellent |
|      400801 |                   750 | Standard  |
|      401250 |                   750 | Standard  |
|      401890 |                  1750 | Good      |
|      402120 |                   750 | Standard  |
|      402500 |                  NULL | Standard  |
|      403000 |                  6000 | Excellent |
|      404150 |                  3500 | High      |
|      404180 |                  3500 | High      |
```

In the first demo we had the case expression

```
        CASE
             WHEN catg_id ='PET'   THEN 0.95
             WHEN catg_id ='SPG'   THEN 0.95
             WHEN catg_id ='APL'   THEN 0.90
             ELSE  1 END
```

You can use other tests in a case expression- In lists

```
          CASE
               WHEN catg_id in('PET', 'SPG')   THEN 0.95
               WHEN catg_id in('APL')   THEN 0.90
               ELSE  1 END
```

AND, OR

```
          CASE
               WHEN catg_id ='PET' or catg_id = 'SPG'   THEN 0.95
               WHEN catg_id ='APL'   THEN 0.90
               ELSE  1 END
```

Some of the other demos used a Between test or a less than test.

2. Simple Case expression.

MySQL has another version of the Case expression called a simple Case expression that uses only equality testing. .

Demo 10:   Simple case; only one attribute is being compared; the comparisons are all equality tests.

```
select  catg_id, prod_id, prod_list_price
, CASE catg_id
     WHEN 'PET'   THEN 0.95
     WHEN 'SPG'   THEN 0.95
     WHEN 'APL'   THEN 0.90
  ELSE  1
  END  * prod_list_price AS "Today's Price"
from product.products
;
```

Selected rows

```
+---------+---------+-----------------+---------------+
| catg_id | prod_id | prod_list_price | Today's Price |
+---------+---------+-----------------+---------------+
| HW      |    1000 |          125.00 |      125.0000 |
| SPG     |    1010 |          150.00 |      142.5000 |
| SPG     |    1020 |           12.95 |       12.3025 |
| HW      |    1110 |           49.99 |       49.9900 |
| APL     |    1120 |          549.99 |      494.9910 |
| APL     |    1125 |          500.00 |      450.0000 |
| APL     |    1130 |          149.99 |      134.9910 |
| PET     |    1140 |           14.99 |       14.2405 |
| PET     |    1141 |           99.99 |       94.9905 |
| APL     |    4569 |          349.95 |      314.9550 |
```

Demo 11:   Organizing sales by season.

```
select  order_id, date_format(order_date, '%Y/%m/%d') AS OrderDate
, CASE quarter(order_date)
     WHEN 1   THEN 'winter'
     WHEN 2   THEN 'spring'
     WHEN 3   THEN 'summer'
     WHEN 4   THEN 'fall'
  END   AS "Season"
from orderEntry.OrderHeaders ;
```

Selected rows

```
+----------+------------+--------+
| order_id | OrderDate  | Season |
+----------+------------+--------+
|      105 | 2015/10/01 | fall   |
|      106 | 2015/10/01 | fall   |
|      107 | 2015/10/02 | fall   |
|      109 | 2015/10/12 | fall   |
|      223 | 2016/03/05 | winter |
|      301 | 2015/06/04 | spring |
|      302 | 2015/06/04 | spring |
|      307 | 2015/06/04 | spring |
```

Demo 12: Using a case to do a special sort. We want to sort the products by the categories but not alphabetically. The order we want to use is PET, SPG, APL, HW.

```
select catg_id, prod_id, prod_list_price
from product.products
order by CASE catg_id
            WHEN 'PET'   THEN 1
            WHEN 'SPG'   THEN 2
            WHEN 'APL'   THEN 3
            WHEN 'HW'    THEN 4
         ELSE  9999
         END,
         catg_id, prod_id;

selected rows
+---------+---------+-----------------+
| catg_id | prod_id | prod_list_price |
+---------+---------+-----------------+
| PET     |    1140 |           14.99 |
| PET     |    1141 |           99.99 |
| PET     |    1142 |            2.50 |
| PET     |    1150 |            4.99 |
| PET     |    1151 |           14.99 |
| SPG     |    1050 |          269.95 |
| SPG     |    1060 |          255.95 |
| APL     |    1120 |          549.99 |
| APL     |    1125 |          500.00 |
| APL     |    1126 |          850.00 |
| HW      |    1000 |          125.00 |
| HW      |    1070 |           25.50 |
| HW      |    1071 |           25.50 |
| GFD     |    5000 |           12.50 |
| GFD     |    5001 |            5.00 |
| HD      |    5002 |           23.00 |
| HD      |    5004 |           15.00 |
| HD      |    5005 |           45.00 |
| HD      |    5008 |           12.50 |
+---------+---------+-----------------+
```