#### **CASE AND IT'S USE CASE**

The CASE statement is SQL's way of handling if/then logic. The CASE statement is followed by at least one pair of WHEN and THEN statements—SQL's equivalent of IF/THEN in Excel. Because of this pairing, you might be tempted to call this SQL CASE WHEN, but CASE is the accepted term

#### **USE CASE 1**

 Use CASE to COUNT the number of rows in a column match a condition

**Results:** 

Slide for the next page



#### ItemsCount ExpensiveItemsCount

5 3

Follow for more Questions
Siddharth Nigam

#### **CASE AND IT'S USE CASE**

#### **USE CASE 2**

• Searched CASE in SELECT (Matches a boolean expression)

```
SELECT Id, ItemId, Price,

CASE WHEN Price < 10 THEN 'CHEAP'

WHEN Price < 20 THEN 'AFFORDABLE'

ELSE 'EXPENSIVE'

END AS PriceRating

FROM ItemSales
```

### Id ItemId Price PriceRating

1	100	34.5	EXPENSIVE
2	145	2.3	CHEAP
3	100	34.5	EXPENSIVE
4	100	34.5	EXPENSIVE
5	145	10	AFFORDABLE

Slide for the next page





#### **CASE AND IT'S USE CASE**

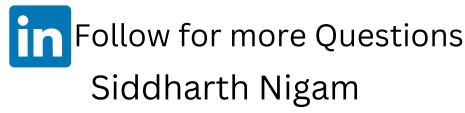
#### **USE CASE 3**

CASE in a clause ORDER BY

```
SELECT * FROM DEPT
ORDER BY
CASE DEPARTMENT
WHEN 'MARKETING' THEN 1
WHEN 'SALES' THEN 2
WHEN 'RESEARCH' THEN 3
WHEN 'INNOVATION' THEN 4
ELSE 5
END,
CITY
```

ID	REGION	CITY	DEPARTMENT	EME	PLOYEES_NUMBER
12	New England	Boston	MARKETING	9	
15	West	San Francisco	MARKETING	12	
9	Midwest	Chicago	SALES	8	
14	Mid-Atlantic	New York	SALES	12	
5	West	Los Angeles	RESEARCH	11	
10	Mid-Atlantic	Philadelphia	RESEARCH	13	
4	Midwest	Chicago	INNOVATION	11	
2	Midwest	Detroit	<b>HUMAN RESOURCES</b>	9	Slide for the next p





### **CASE AND IT'S USE CASE**

#### **USE CASE 4**

Shorthand CASE in SELECT

```
SELECT Id, ItemId, Price,
CASE Price WHEN 5 THEN 'CHEAP'
WHEN 15 THEN 'AFFORDABLE'
```

```
ELSE 'EXPENSIVE'
END as PriceRating
FROM ItemSales
```

Slide for the next page

Follow for more Questions

Siddharth Nigam

#### **CASE AND IT'S USE CASE**

### **USE CASE 5**

Using CASE in UPDATE

```
UPDATE ItemPrice
SET Price = Price *
  CASE ItemId
    WHEN 1 THEN 1.05
    WHEN 2 THEN 1.10
    WHEN 3 THEN 1.15
    ELSE 1.00
  END
```

Slide for the next page

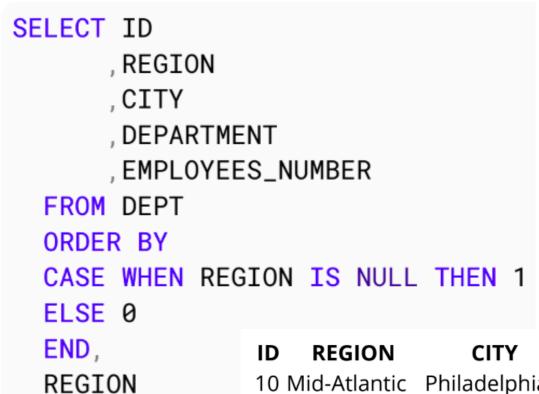
Follow for more Questions

Siddharth Nigam

### **CASE AND IT'S USE CASE**

### **USE CASE 6**

CASE use for NULL values ordered last



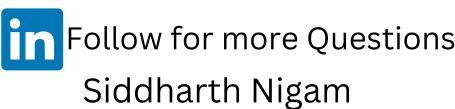
in this way 'O' representing the known values are ranked first, '1' representing the NULL values are sorted by the last:



ID	REGION	CITY	DEPARTMENT	EMPLOYEES_NUMBER
10	Mid-Atlantic	Philadelphia	RESEARCH	13
14	Mid-Atlantic	New York	SALES	12
9	Midwest	Chicago	SALES	8
12	New England	Boston	MARKETING	9
5	West	Los Angeles	RESEARCH	11
15	NULL	San Francisco	MARKETING	12
4	NULL	Chicago	INNOVATION	11
2	NULL	Detroit	HUMAN RESOURCES	9

Slide for the next page





#### **CASE AND IT'S USE CASE**

#### **USE CASE 7**

 CASE in ORDER BY clause to sort records by lowest value of 2 columns

#### Sample data

#### Date2 Date1 1 2017-01-01 2017-01-31 2 2017-01-31 2017-01-03 3 2017-01-31 2017-01-02 4 2017-01-06 2017-01-31 5 2017-01-31 2017-01-05 6 2017-01-04 2017-01-31

#### Query

```
Date1
               Date2
1 2017-01-01 2017-01-31
3 2017-01-31 2017-01-02
2 2017-01-31 2017-01-03
6 2017-01-04 2017-01-31
5 2017-01-31 2017-01-05
4 2017-01-06 2017-01-31
```

## **Explanation** $\downarrow$

As you see row with Id = 1 is first, that because Date1 have lowest record from entire table 2017-01-01, row where Id = 3 is second that because Date2 equals to 2017-01-02 that is second lowest value from table and so on. So we have sorted records from 2017-01-01 to 2017-01-06 ascending and no care on which one column Date1 or Date2 are those values.

```
SELECT Id, Date1, Date2
FROM YourTable
ORDER BY CASE
           WHEN COALESCE(Date1, '1753-01-01') < COALESCE(Date2, '1753-01-01') THEN Date1
           ELSE Date2
         END
```

#### Results

Slide for the next page

Follow for more Questions Siddharth Nigam