A CASE statement is an extended version of the DECODE statement that can accept multiple expressions and return different sets of results.

- **Syntax:**
  
  ```sql
  CASE selector (optional) 
  WHEN condition THEN statement 
  WHEN condition THEN statement 
  ELSE statement (optional) 
  END
  
  “When selector equals condition then result (else default)”
  ```

- **Example:**
  
  ```sql
  CASE FGBTRNH.FGBTRNH_DR_CR_IND 
  WHEN ‘C’ THEN -FGBTRNH.FGBTRNH_TRANS_AMT 
  WHEN ‘D’ THEN FGBTRNH.FGBTRNH_TRANS_AMT, 
  ELSE ‘0’ 
  END
  
  The above statement reads “When debit/credit indicator equals credit, then return the negative of Trans Amt. When debit/credit indicator equals debit then just return the (positive) Trans Amt. If debit/credit indicator equals anything else, return 0 as the default.”
  ```

- **Selector:**
  
  The selector can be a bind variable that prompts the user for the value used in the When statements.
### CASE Statement

- **Example of bind variable as selector**

```sql
CASE :cycle
  WHEN '01' THEN FGBOPAL.FGBOPAL_01_YTD_ACTV
  WHEN '02' THEN FGBOPAL.FGBOPAL_02_YTD_ACTV
  WHEN '03' THEN FGBOPAL.FGBOPAL_03_YTD_ACTV
  etc........
END
```

The above statement reads “**When** the user enters 01 as the cycle, **then** return cycle 1 YTD figure. **When** the user enters 02 as the cycle, **then** return cycle 2 YTD figure...”

- **To use more complex boolean conditions, leave out the optional selector.** For example:

```sql
CASE
  WHEN FTVORG_LEVELS.LEVEL6 LIKE 'D%'
    THEN FTVORG_LEVELS.LEVEL6
  WHEN FTVORG_LEVELS.LEVEL7 LIKE 'D%'
    THEN FTVORG_LEVELS.LEVEL7
  ELSE 'ERROR'
END
```

The above statement reads “**When** level6 starts with a D **then** return level6. **When** level7 starts with a D **then** return level7. If either of these conditions do not match, return the word ‘ERROR’”.

- **Keep in mind that once the condition is met, the CASE statement will stop.** Order the when/then statements in a way that make sense for what you are trying to accomplish. When using variable boolean expressions in the **When** statements, order them from most restrictive to least restrictive to accurately group resulting values.