Using Conversion Functions and Conditional Expressions

TO_CHAR with dates

TO_CAHR(date[,'format_model'])

The format model:

- Must be enclosed with single quotation marks
- Is case-sensitive
- Can include any valid date format element
- has an fm element to remove padded blanks or suppress leading zeros
- Is separated from the date value by a comma

TO_CHAR with numbers

TO_CAHR(number[,'format_model'])

These are some of the format elements that you can use with the TO_CHAR function to display a number value as a character:

Element	Result
9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a comma as a thousands indicator

NVL, NVL2, NULLIF and COALESCE Functions

The following functions work with any data type and pertain to using nulls:

- NVL (expr1, expr2)
- NVL2 (expr1, expr2, expr3)
- NULLIF (expr1, expr2)
- COALESCE (expr1, expr2, ..., exprn)

Funciton	Description
NVL	Converts a null value to an actual value
NVL2	If expr1 is not null, NVL2 returns expr2. If expr1 is null, NVL2 returns expr3. The argument expr1 can have any data type
NULLIF	Compares two expressions and returns null if they are equal; returns the first expression if they are not equal
COALESCE	Returns the first non-null expression in the expression list

The CASE Function

Conditional Expression

- Provide the use of the IF-THEN-ELSE logic within a SQL statement
- Use two methods:
  - CASE expression

- DECODE function

The two methods that are used to implement conditional processing (IF-THEN-ELSE logic) in a SQL statement are the CASE expression and the DECODE function

Note: The CASE expression complies with the ANSI SQL. The DECODE function is specific to Oracle syntax.

CASE Expression

Facilitates conditional inquires by doing the work of an IF-THEN-ELSE statement:

```
CASE expr WHEN comparison_expr1 THEN return_expr1
         [WHEN comparison_expr2 THEN return_expr2
          WHEN comparison_exprn THEN return_exprn
          ELSE else_expr]
END
```

Examples:

```
SELECT first_name, job_id, salary,
    CASE job_id WHEN 'IT_PROG' THEN '1.10*salary
                WHEN 'ST_CLERK' THEN 1.15*salary
                WHEN 'SA_REP' THEN 1.20*salary
    ELSE salary
    END 'REVISED_SALARY'
FROM EMPLOYEES;
```

FIRST_NAME	JOB_ID	SALARY	REVISED_SALARY
Steven	AD PRES	24000	24000
Neena	AD VP	17000	17000
Alexander	IT PROG	9000	9900

More flexible method:

```
SELECT first_name, job_id, salary,
    CASE WHEN JOB_ID = 'IT_PROG' THEN '1.10*salary
         WHEN JOB_ID = 'ST_CLERK' THEN 1.15*salary
         WHEN JOB_ID = 'SA_REP' THEN 1.20*salary
    ELSE salary
    END 'REVISED_SALARY'
FROM EMPLOYEES;
```

If you didn't put ELSE statement, then null will appear for unmatched conditions

```
SELECT first_name, job_id, salary,
    CASE job_id WHEN 'IT_PROG' THEN '1.10*salary
                WHEN 'ST_CLERK' THEN 1.15*salary
                WHEN 'SA_REP' THEN 1.20*salary
    END 'REVISED_SALARY'
FROM EMPLOYEES;
```

FIRST_NAME	JOB_ID	SALARY	REVISED_SALARY
Steven	AD PRES	24000	(null)
Neena	AD VP	17000	(null)
Alexander	IT PROG	9000	9900

This below statement is not logically correct.
If the first condition is met, then it will show the result regardless of another conditions.

```
SELECT salary,
CASE WHEN salary > 3000 THEN 'salary > 3000'
     WHEN salary > 4000 THEN 'salary > 4000'
     WHEN salary > 10000 THEN 'salary > 10000'
END FFF
FROM EMPLOYEES;
```

SALARY	FFF
24000	salary > 3000
17000	salary > 3000
17000	salary > 3000
9000	salary > 3000

To fix the problem, we need to start with the highest number.

```
SELECT salary,
CASE WHEN salary > 10000 THEN 'salary > 10000'
     WHEN salary > 4000 THEN 'salary > 4000'
     WHEN salary > 3000 THEN 'salary > 3000'
```

```
END FFF
FROM EMPLOYEES;
```

SALARY	FFF
24000	salary > 10000
17000	salary > 10000
17000	salary > 10000
9000	salary > 4000

The DECODE Function

Facilitates conditional inquiries by doing the work of a CASE expression or an IF-THEN-ELSE statement:

```
DECODE (col | expression, search1, result1
                          [, search2, result2, ...,]
                          [, default])
```

```
SELECT first_name, job_id, salary,
    DECODE(job_id, 'IT_PROG', '1.10*salary,
                 ,'ST_CLERK', 1.15*salary,
                 , 'SA_REP', 1.20*salary,
            salary)
    REVISED_SALARY
FROM EMPLOYEES;
```

If you didn't put default vlaue for unmatched conditions, then null will be returned for these values.

Example
Display tax for employees as follow:
If salary < 3000, then tax = 0
if 3000 <= salary <= 7000, then tax=10%
if salary > 7000, then tax=20%

We should use CASE instead of DECODE since the CASE is more flexible.

```
SELECT EMPLOYEE_ID, FIRST_NAME, SALARY
```

```
    CASE WHEN SALARY < 3000 THEN '0%
    WHEN SALARY BETWEEN 3000 AND 7000 THEN '10%'
    WHEN SALARY > 7000 THEN '20%'
END TAX
FROM EMPLOYEES;
```