USING CONDITIONAL EXPRESSIONS

General Functions

General functions are used to handle NULL values in database. The objective of the general NULL handling functions is to replace the NULL values with an alternate value. We shall briefly see through these functions below.

NVL

The NVL function substitutes an alternate value for a NULL value.

Syntax:

```
NVL( Arg1, replace_with )
```

In the syntax, both the parameters are mandatory. Note that NVL function works with all types of data types. And also that the data type of original string and the replacement must be in compatible state i.e. either same or implicitly convertible by Oracle.

If arg1 is a character value, then oracle converts replacement string to the data type compatible with arg1 before comparing them and returns VARCHAR2 in the character set of expr1. If arg1 is numeric, then Oracle determines the argument with highest numeric precedence, implicitly converts the other argument to that data type, and returns that data type.

The SELECT statement below will display 'n/a' if an employee has been not assigned to any job yet i.e. JOB_ID is NULL. Otherwise, it would display the actual JOB_ID value.

```
SELECT  first_name, NVL(JOB_ID, 'n/a')
FROM employees;
```

NVL2

As an enhancement over NVL, Oracle introduced a function to substitute value not only for NULL columns values but also for NOT NULL columns. NVL2 function can be used to substitute an alternate value for NULL as well as non NULL value.

Syntax:

```
NVL2( string1, value_if_NOT_null, value_if_null )
```

The SELECT statement below would display 'Bench' if the JOB_CODE for an employee is NULL. For a definite not null value of JOB CODE, it would show constant value 'Job Assigned'.

```
SQL> SELECT NVL2(JOB_CODE, 'Job Assigned', 'Bench')
FROM employees;
```

NULLIF

The NULLIF function compares two arguments expr1 and expr2. If expr1 and expr2 are equal, it returns NULL; else, it returns expr1. Unlike the other null handling function, first argument can't be NULL.

Syntax:

```
NULLIF (expr1, expr2)
```

Note that first argument can be an expression that evaluates to NULL, but it can't be the literal NULL. Both the parameters are mandatory for the function to execute.

The below query returns NULL since both the input values, 12 are equal.

```
SELECT NULLIF (12, 12)
FROM DUAL;
```

Similarly, below query return 'SUN' since both the strings are not equal.

```
SELECT NULLIF ('SUN', 'MOON')
FROM DUAL;
```

COALESCE

COALESCE function, a more generic form of NVL, returns the first non-null expression in the argument list. It takes minimum two mandatory parameters but maximum arguments has no limit.

Syntax:

```
COALESCE (expr1, expr2, ... expr_n )
```

Consider the below SELECT query. It selects the first not null value fed into address fields for an employee.

```
SELECT COALESCE (address1, address2, address3) Address
FROM  employees;
```

Interestingly, the working of COALESCE function is similar to IF..ELSIF..ENDIF construct. The query above can be re-written as -

```
IF address1 is not null THEN
   result := address1;
ELSIF address2 is not null THEN
   result := address2;
ELSIF address3 is not null THEN
   result := address3;
ELSE
   result := null;
END IF;
```

Conditional Functions

Oracle provides conditional functions DECODE and CASE to impose conditions even in SQL statement.

The DECODE function

The function is the SQL equivalence of IF..THEN..ELSE conditional procedural statement. DECODE works with values/columns/expressions of all data types.

Syntax:

```
DECODE (expression, search, result [, search, result]... [, default])
```

DECODE function compares expression against each search value in order. If equality exists between expression and search argument, then it returns the corresponding result. In case of no match, default value is returned, if defined, else NULL. In case of any type compatibility mismatch, oracle internally does possible implicit conversion to return the results.

As a matter of fact, Oracle considers two nulls to be equivalent while working with DECODE function.

```
SELECT DECODE(NULL,NULL,'EQUAL','NOT EQUAL')
FROM DUAL;

DECOD
-----
EQUAL
```

If expression is null, then Oracle returns the result of the first search that is also null. The maximum number of components in the DECODE function is 255.

```
SELECT first_name, salary, DECODE (hire_date, sysdate,'NEW JOINEE','EMPLOYEE')
 FROM employees;
```

CASE expression

CASE expressions works on the same concept as DECODE but differs in syntax and usage.

Syntax:

```
CASE  [ expression ]
   WHEN condition_1 THEN result_1
   WHEN condition_2 THEN result_2
   ...
   WHEN condition_n THEN result_n
   ELSE result
END
```

Oracle search starts from left and moves rightwards until it finds a true condition, and then returns result expression associated with it. If no condition is found to be true, and an ELSE clause exists, then Oracle returns result defined with else. Otherwise, Oracle returns null.

The maximum number of arguments in a CASE expression is 255. All expressions count toward this limit, including the initial expression of a simple CASE expression and the optional ELSE expression. Each WHEN ... THEN pair counts as two arguments. To avoid exceeding this limit, you can nest CASE expressions so that the return_expr itself is a CASE expression.

```
SELECT first_name, CASE WHEN salary < 200 THEN 'GRADE 1'
   WHEN salary > 200 AND salary < 5000 THEN 'GRADE 2'
   ELSE 'GRADE 3'
     END CASE
FROM employees;

ENAM    CASE
----    -------
JOHN    GRADE 2
EDWIN   GRADE 3
KING    GRADE 1
```