SINGLE ROW FUNCTIONS IN SQL

1. Introduction to Oracle Built-in Functions

Oracle Built-in Functions are the functions supplied by Oracle that can be used to manipulate data items and return a result.

There are two types of Built-in functions available in Oracle.

- **Single Row Functions:** Single row or Scalar functions return a value for every row that is processed
- **Group Functions:** These functions group the rows of data based on particular column values, performs the aggregate function (sum, avg etc.) on each group and return one result row per group of rows

2. Dual table

- Dual table is a single row and single column dummy table provided by Oracle. This is used to perform mathematical calculations without using a user defined table
- Oracle presents the output of every operations in a tabular format so that it seems to the user that the output comes from a user defined table

Default content of dual table	Sample output using dual table
Selects * from DUAL;	Select 20*23 from DUAL;
DUMMY -------------- X	20*23 ------------- 460

3. Single Row Functions

Four different types of single row functions are as follows:

a) **Conversion Functions:** These are functions that help us to convert a value in one form to another form. For Example: a null value into an actual value, or a value from one data type to another data type like NVL, TO_CHAR, TO_NUMBER, TO_DATE etc.

b) **Character or Text Functions:** These are functions that accept character input and can return both character and number values.

c) **Numeric Functions:** These are functions that accept numeric input and return numeric values.

d) **Date Functions:** These are functions that take values that are of data type DATE as input and return values of data type DATE, except for the MONTHS_BETWEEN function, which returns a number.

Consider the following table Employee:

EMP_ID	EMP_NAME	SALARY	DEP_NAME	BRANCH_NAME
1	amit	10000	HR	Kolkata
2	ajay	20000	Marketing	Mumbai
3	sima	16000	HR	Ahmedabad
4	dipa	40000	Admin	Kolkata
5	anuj		Marketing	Ahmedabad

4. Conversion Functions

- NVL
- NVL2
- NULLIF

NVL: To replace the null in a result set with a string

Syntax : `NVL( string1, replace_with)`

If `string1` is null, then `NVL` returns `replace_with`.
If `string1` is not null, then `NVL` returns `string1`.

Example:
`SELECT NVL(emp_name,'NA') FROM Employee;`
The above query will display 'NA' wherever emp_name is null.
We can also replace with another column.
Example: `SELECT NVL(emp_name,dep_name) FROM Employee;`

The emp_name and dep_name should belongs to same data type family.
The above query will display dep_name wherever emp_name is NULL.

Example: `SELECT NVL(salary,0) FROM Employee;`
The above query returns 0 only if the salary is defined as NUMBER and is NULL.

NVL2: NVL2 function extends the functionality found in the NVL Function.

It allows to substitute a value when a null value is encountered as well as when a non-null value is encountered.

Syntax : `NVL2(string1,value_if_not_null,value_if_null)`

if string1 is not null then NVL2 returns value_if_not_null.
if string1 is null then NVL2 returns value_if_null.

Example: `SELECT NVL2(emp_name,dep_name,'NOT AVAILABLE') FROM Employee;`

NULLIF: `NULLIF` compares *expr1* and *expr2*. If they are equal, then the function returns null. If they are not equal, then the function returns *expr1*.

Syntax : `NULLIF(expr1,expr2)`

Example: `SELECT NULLIF( dep_name,'HR')  FROM Employee;`

The above query returns NULL if dep_name is 'HR', else it will return the dept_name.
Note: You cannot specify the literal `NULL` for *expr1*.

5. Character Functions

Different types of character functions are

- Character to character functions accept string as input and will give string as output

  - INITCAP
  - LOWER
  - UPPER
  - CONCAT
  - LPAD,RPAD
  - TRIM
  - SUBSTR
  - REPLACE

- Character to number functions accept string as input and will give number as output

  - LENGTH
  - INSTR

a) **INITCAP:** Sets the first character in each word to upper case and the rest to lower case.

Syntax : `INITCAP(expr1)`

Example: `SELECT INITCAP(emp_name) FROM Employee;`

```
Amit
Ajay
Sima
Dipa
Anuj
```

The above query returns all the employee names with the first letter in upper case and rest of the characters in lower case.

b) **LOWER:** This function converts all letters in the specified string to lower case. If there are characters in the string that are not letters, they are unaffected by this function.

Syntax : `LOWER(expr1)`

Example: `SELECT LOWER (emp_name) FROM employee;`

```
amit
ajay
sima
dipa
anuj
```

The above query returns all the characters of the employee name in lower case.

c) **UPPER:** This function converts all letters in the specified string to upper case. If there are characters in the string that are not letters, they are unaffected by this function.

Syntax : `UPPER(expr1)`

Example: `SELECT UPPER (emp_name) FROM Employee;`

```
AMIT
AJAY
SIMA
DIPA
ANUJ
```

The above query returns all the characters of the employee name in upper case.

d) **CONCAT:** This function allows you to concatenate two strings together.

Syntax : `CONCAT(expr1,expr2)`

Example: `SELECT CONCAT(emp_name,dep_name) full_name FROM Employee;`

The above query returns the emp_name & dep_name concatenated into a single string.

e) **SUBSTR:** Returns specified characters from a string, starting from specific position 'm' to required characters length 'n'.

Syntax : `SUBSTR(col/expr,m,n)`

If 'm' is positive, Oracle counts from beginning of string, If it is negative Oracle counts from the end of string.
If 'n' is omitted, Oracle returns all characters to end of the string starting from m.

Example: `SELECT SUBSTR(emp_name,3,2) FROM Employee;`

The above query starts searching from the third position of the employee name, from the starting of the string and displays two characters from there.

Example: `SELECT SUBSTR('abcdefg',-4,2) FROM dual;`
Output: `de`
The above query starts searching from the fourth position of the given string, from the end of the string and display two characters from there.

f) **REPLACE:** It returns every occurrence of search_string replaced by the replacement_string.If the replacement string is omitted or null all occurrences of search string will be removed.

Syntax : REPLACE(string,search_string,replace_string)

Example:
```
SELECT  REPLACE(branch_name,'Mumbai','Kolkata') FROM
Employee WHERE dep_name = 'HR';
```

The above query replaces branch name to 'Kolkata' wherever 'Mumbai' is available for the HR department.

g) **LPAD,RPAD:**
LPAD pads the character value right justified to a total width of n character positions.

Syntax : `LPAD(expr1,padded_length,padded_string)`

RPAD pads the character value left justified to a total width of n character positions.

Syntax : `RPAD(expr1,padded_length,padded_string)`

The default padding character is space.

Example1: `SELECT LPAD('jhon',8 ,'x') FROM dual;`
Output : `xxxxjhon`
The above query fills the four blank spaces with 'x' left of the given string.

Example2: `SELECT LPAD('jhon',8) FROM dual;`
Output : `    jhon`
Since the third parameter is not specified, result of the above query will be by default space padded for the previous four positions.

Example3: `SELECT RPAD('jhon',8 ,'x') FROM dual;`
Output :`jhonxxxx`
The above query fills the four blank spaces with 'x' right of the given string.

h) **TRIM:** It enables to trim leading or trailing characters or both from a string. If we don't specify anything, it will trim spaces.

Syntax :

`[LTRIM/RTRIM](text_string,trim_character)`

where trim_character is the character that is to be trimmed from the text_string.

TRIM(character FROM string)

LTRIM : Removes the leading characters
RTRIM : Removes the trailing characters
TRIM : Removes both

Example1: `SELECT RTRIM('ssmithss','s') FROM dual;`
Output: `ssmith`
The above query removes 'trailing' 's' from the given string.

Example2: `SELECT LTRIM('ssmithss','s') FROM dual;`
Output : `mithss`
The above query removes 'leading' 's' from the given string.

Example3: `SELECT TRIM('s' from 'ssmiths') FROM dual;`
Output: `mith`
The above query removes 'trailing' & 'leading' 's' from the given string.

Example4: `SELECT TRIM('   smith    ') FROM dual;`

```
Output: smith
The above query removes 'trailing' & 'leading'  spaces from
```

the given string.

i) **INSTR:** This function returns the location of a sub string in a given string.

Syntax : `INSTR( string, sub_string [, start_position [, nth_appearance ] ] )`

start_position and nth_appearance are optional. If not specified, always INSTR starts with first position and will give first appearance.

Example1: `SELECT INSTR('internet','e')  FROM dual;`
Output: 4
The above query returns the first position of 'e' searched from the start of the given string.

Example2: `SELECT INSTR('internet','e',1,2) FROM dual;`
Output: 7
The above query returns the second position of 'e' searched from the start of the given string.

Example3: `SELECT INSTR('internet','e',5,1) FROM dual;`
Output : 7
The above query returns the first position of 'e' searched from the fifth position of the given string.

j) **LENGTH:** Returns number of characters in a value.

Syntax : `LENGTH(column)`

Example1: `SELECT LENGTH(branch_name) FROM Employee;`

The above query returns number of characters in the branch_name field for each and every record.

Example2: `SELECT LENGTH('jhon') FROM dual;`

```
Output:  4
```

The above query returns the number of characters from the given string.

6. Numeric Functions

- Numeric functions are used to perform operations on numbers
- They accept numeric values as input and return numeric values as output

- Following are the few examples of Numeric functions available in Oracle

  - ABS and MOD
  - POWER and SQRT
  - FLOOR and CEIL
  - TRUNC and ROUND

a) ABS and MOD Function

The ABS function returns the absolute value of the parameter passed.

Syntax : `ABS(number)`

Example: `SELECT ABS(-10) FROM dual;`
Output: `10`
The above query returns the absolute value of the given 'number'.

The MOD function returns the remainder value of the parameter passed.

Syntax : `MOD(number1,number2)`

Example: `SELECT MOD(10,4) FROM dual;`
Output: `2`
The above query returns the remainder when 10 is divided by 4.

b) POWER and SQRT Function

POWER function returns the argument raised to the specified power.

Syntax : `POWER(number1,number2)`

Example: `SELECT POWER(4,3) As Cube FROM dual;`
Output: `64`
The above query returns the output when 4 is raised to the power of 3.

SQRT function returns the square root of a number passed as parameter.

Syntax: `SQRT(number)`

Example: `SELECT SQRT(64) As SquareRoot FROM dual;`
Output : `8`
The above query returns the square root value of 64.

c) FLOOR and CEIL Function

The FLOOR function returns the largest integer less than or equal to the value passed in parameter.

Syntax: **`FLOOR(decimal number)`**

Example1: `SELECT FLOOR(7.14), FLOOR(7.84) FROM dual;`

Output: `7          7`

The above query returns the largest integer nearest to 7.14 & 7.84.

Example2: `SELECT FLOOR(-7.14) FROM dual;`

Output : `-8`

The above query returns the largest integer nearest to -7.14.

The CEIL function returns the smallest integer greater than or equal to the value mentioned in parameter.

Syntax: **`CEIL(decimal number)`**

Example1: `SELECT CEIL(7.14), CEIL(7.84) FROM dual;`

Output: `8         8`

The above query returns the smallest integer nearest to 7.14 & 7.84.

Example2: `SELECT CEIL(-7.14) FROM dual;`

Output: `-7`

The above query returns the smallest integer nearest to -7.14.

d) TRUNC and ROUND Function

- The TRUNC function truncates the value present in the column, expression up to decimal places mentioned in first parameter.
- If the second argument is 0 or is missing, the value is truncated to zero decimal places.

Syntax: **`TRUNC(decimal number,number of places)`**

Example1: `SELECT TRUNC(137.5738,3) As Rounded FROM dual;`

Output: `137.573`

The above query returns the decimal number with three digits after the decimal point.

Example2: `SELECT TRUNC(137.5738,0) As Rounded FROM dual;`

Output: `137`

The above query returns the integer value.

- The ROUND function round off the value present in the column, expression up to decimal places mentioned in first parameter.
- If the second argument is 0 or is missing, the value is rounded to zero decimal places.

Syntax: **`ROUND(decimal number,number of places)`**

Example 1: `SELECT ROUND(137.5738,3) As Rounded FROM dual;`

Output: `137.574`

The above query returns the decimal number with three digits after the decimal point where 4th digit is rounded.

Example 2: `SELECT ROUND(137.5738,0) As Rounded FROM dual;`

Output: `138`

The above query returns the integer value rounded to the next highest value.

If the second argument is negative number, the value is rounded up specified decimal places to the left (rounded to the nearest unit of 10).

Example 3: `SELECT ROUND(137.5748,-1) As Rounded FROM dual;`

Output: `140`

The above query returns the integer value which is the nearest tens value.

Example 4: `SELECT ROUND(137.5748,-2) As Rounded FROM DUAL;`

Output: `100`

The above query returns the integer value which is the nearest hundreds value.