-Single row functions: Functions that work on a single row once at a time.

-CONCAT (‘<string1>’, ‘<string2>’) -> works as ‘<string1>’ || ‘<string2>’

-UPPER (‘<string>’) -> makes the string all upper case.

NOTE: We know that the single row functions work on every single row one at time. So it’s totally valid to write:

SELECT UPPER(‘hello’)

FROM <some\_table>

And this will print out:

HELLO

HELLO

HELLO

…

-NOTE: There is a special table in Oracle that is called the DUAL table. So if we want to example test a function for example UPPER we can write.

SELECT UPPER(‘hello’)

FROM DUAL

And this will print out:

HELLO

That is because the DUAL table has only one cell.

This DUAL table is very good for testing functionality as well, cause even though it has only one cell, we can easily make so that we change the column name and the cell value. We can also make it have multiple columns.

Example:

SELECT \*

FROM DUAL

Will print out:

DUMMY -> column name

X -> cell

But if we write for example something like:

SELECT ‘pizza’ as FOOD

FROM DUAL

It will print out:

FOOD -> column name

pizza -> cell

We can also write:

SELECT ‘pizza’ as FOOD, ‘fanta’

FROM DUAL

And this will print out

FOOD | ‘FANTA’ -> column names

pizza | fanta -> cells value

-MORE on Single Row Functions:

-LOWER(‘<string>’) -> makes the strings lower case.

-INITCAP(‘<string>’) -> makes every new word start with capital and other characters, lower.

-LENGTH(‘<string>’) -> number of characters in the string INCLUDING the spaces.

-SUBSTR(‘<string>’, <starting\_pos>, <number\_of\_characters>):

-> Example: SUBSTR(‘HELLO’, 2, 2) will print EL

-LPAD(‘<string>’, <width\_we\_want\_to\_be\_the\_data>, ‘<char>’)

->Creates a padding on the left of the string based on the length we want it to be.

->Example: LPAD(‘hello’, 6, ‘&’) will print out &hello

-RPAD ->works as the LPAD

-LTRIM(‘<string>’, ‘<char>’) trims the left side of the string and takes out all the left ‘<char>’ until met with character that is not ‘<char>’.

-> Example: LTRIM(‘hhhhhhello’, ‘h’) will print ello

-RTRIM ->works as LTRIM

NOTE: Great website to check all the functions in SQL is: <https://www.techonthenet.com/oracle/index.php>

- Since now we have used the single row functions in the SELECT clause, but we can use them in the WHERE clause as well.

For example:

SELECT ENAME

FROM EMP

WHERE ENAME = upper(‘martin’)

NOTE: All the functions we have talked so far are Character functions.

-Numeric functions:

-ROUND(number, <decimal\_we\_want\_to\_show>)

-> ROUND(100.7088, 2) = 100.71 (rounds mathematically)

NOTE: ROUND(number) will round to whole number.

-TRUNC(number, <decimal\_we\_want\_to\_show>) Like ROUND but it just shows the decimal numbers, it doesn’t round mathematically

->Example: TRUNC(10.9999, 2) will print 10.99

NOTE: again we can see the numeric functions in:

<https://www.techonthenet.com/oracle/index.php>

-Date functions

-SYSDATE -> shows the date of the system that is installed.

-SYSTIMESTAMP ->returns not only the date, but the time and time zone

-ADD\_MONTHS(<date>, <number\_of\_dates\_to\_add>)

->Example: ADD\_MONTHS(’11/17/2012’, 3) will print 02/17/2013

NOTE: we can go back in time by giving a negative argument for the <number\_of\_dates\_to\_add>)

-MONTHS\_BETWEEN(<dateA>, <dateB>) -> gives the months between the two dates.

-we can also use TRUNC for dates

->example: TRUNC(<date>, ‘month’> will show us the first of the month. So for example TRUNC(12/05/2016, ‘MONTH’) will give us 12/01/2016

And TRUNC(12/05/2016, ‘YEAR’) will give us 01/01/2016, where truncing with the year gives us the first of the year.

->example: TRUNC(systimestamp) will give us sysdate

- LAST\_DAY(d)

last\_day is a date function that requires a date as an argument. It returns the last day of the month in which the given date falls. The argument is required for this function to work properly.

- NEXT\_DAY(d, c)

The first argument is the date and the second argument is a text reference to a day of the week. Both arguments are required for this function to work properly. This function returns a valid date representing the first occurrence of the c day following the date represented in d.

-Conversion functions

-TO\_CHAR(sysdate, ‘MMM-DD-YYYY) ->Result: 05-31-2016

Or TO\_CHAR(sysdate, ‘MONTH DD, YYYY) -> Result: May 31, 2016

NOTE: Here are the abbreviation

YEAR -> Full name of the year

MM -> two digit value of the month

MONTH -> Full name of the month

MON -> Three letter abbreviation of the month

DY -> Three letter abbreviation of the day

DAY -> Full name of the day

DD -> Two digit value of the day

YYYY -> four digit for the year

We can also make it reader friendly the result of the print as well:

SELECT TO\_CHAR(sysdate, ‘ddth “of” month, yyyy’) from dual.

And we can combine the top abbreviations.

-TO\_CHAR(123, ‘$999.99’) ->Result: $123.00

NOTE: Here are what we can write for second argument:

9 -> represents any number:

Example: format usage 9999.99

Input: 27.6 -> Output: 27.60

0 -> Forces a zero in a display:

Example: format usage 0000.00

Input: 278.6

Output: 0278.60

$ ->Puts a dollar sign

. -> Decimal point

, Comma for thousands separator

Example: format usage: 99,999.99

Input: 12400.8

Output: 12,400.80

And we can combine these.

TO\_CHAR(num) (Without the second argument) is going to convert our numeric type to char type or string

-TO\_DATE(‘<string>’, format)

Example: TO\_DATE(‘2012-08-27’, ‘YYYY-MM-DD’)

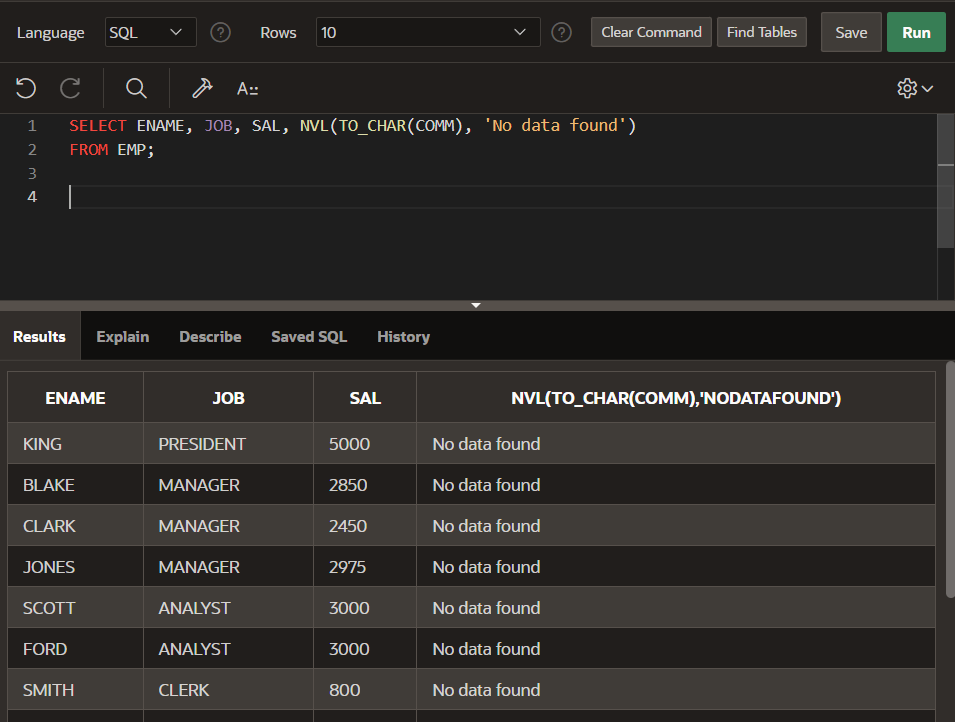
It can work for the next example as well:

TO\_DATE(‘3 of June, 2012’, ‘dd “of” month, yyyy)

-Functions that replace empty data or NULL

-NVL(column, value) -> Everytime it comes across a NULL in a row from that column it replaces it with the value, where value needs to be the same type as the column type.

NOTE!:



-NULLIF(arg1, arg2) Output is null if arg1 = arg2, if arg1 != arg2 then it will return arg1.