READING MATERIAL 5

Changing Column Headings with Column Aliases

As we discussed earlier, Oracle reprints the column name exactly as it was included in the select statement, including functions if there are any. Unfortunately, this method usually leaves you with a bad description of the column data, compounded by the fact that Oracle truncates the expression to fit a certain column length corresponding to the datatype of the column returned. Fortunately, Oracle provides a solution to this situation with the use of column aliases in the select statement. Any column can be given another name by you when the select statement is issued. This feature gives you the ability to fit more descriptive names into the space allotted by the column datatype definition.

SQL> SELECT ename "Employee name", Sal* 12 "Annual Salary" FROM emp;

OUTPUT:

Employee n	**Annual Salary**
SMITH	9600
ALLEN	19200
WARD	15000
JONES	35700
MARTIN	15000
…….	……..

14 rows selected.

**Notice that the column heading in the output is exactly the same as the column alias.

In order to specify an alias, simply name the alias after identifying the column to be selected, with or without an operation performed on it, separated by white space. Alternatively, you can issue the as keyword to denote the alias. By default, alias headings appear in the uppercase. If the alias contains spaces, special characters (such as # or $), or is case sensitive, enclose the alias in the double quotation marks(" ").

SQL> *SELECT ename as "employee name", sal salary FROM emp;*

OUTPUT:

Employee n	**SALARY**
SMITH	9600
ALLEN	19200
WARD	15000
JONES	35700
MARTIN	15000
…….	……..

14 rows selected

Column aliases are useful for adding meaningful headings to output from SQL queries. As shown, aliases can be specified in two ways: either by naming the alias after the column specification separated by white space, or with the use of the as keyword to mark the alias more clearly.

Column Concatenation

Columns can be combined together to produce more interesting or readable output. The method used to merge the output of two columns into one is called concatenation. The concatenation operator is two pipe characters put together, i.e. ||.

In this way we can increase the meaningfulness of the output produced. For good measure, the use of column aliases is recommended in order to make the name of the concatenated columns more meaningful.

Example: In order to display the output in English like sentences we may concatenate two or more columns of the table even with the constant strings as shown below:

SQL> *SELECT 'empno of '||ename ||' is '|| empno FROM emp;*

OUTPUT:

'EMPNOOF'||ENAME||'IS'||EMPNO
empno of SMITH is 7369
empno of ALLEN is 7499
empno of WARD is 7521
……..

14 rows selected

Pattern Matching

In order to select rows that match a particular character pattern we use the LIKE operator. This character matching operation is called as wildcard search. The following symbols are used for matching the pattern % (percentage). This symbol represents any sequence of zero or more characters . __ (underscore) this symbol is used for any single character search. The % and _ symbols can be used in any combination with literal characters.

Pattern matching symbols	**Description**
%(percentage)	represents any sequence of zero or more characters
_(underscore)	it is used for any single character search.

Example: List the employee names and empno whose names start with J.

SQL> SELECT ename, empno FROM emp WHERE ename like 'J%';

OUTPUT:

ENAME	EMPNO
JONES	7566
JAMES	7900

Example: List the employee names whose names are of six characters length.

SQL> *SELECT ename FROM emp WHERE ename like '_ _ _ _ _ _ ';*

OUTPUT:

ENAME
MARTIN
TURNER
MILLER

Example: List the employee names whose names end with ‘h’.

SQL> *SELECT ename FROM emp WHERE ename like ‘%H’;*

Output:

ENAME
SMITH

Example: List the employee names having D as the second character.

SQL> *SELECT ename FROM emp WHERE ename LIKE ‘_D%’;*

OUTPUT:

ENAME
ADAMS

Example: List the employee names having two A in their name.

SQL> *SELECT ename FROM emp WHERE ename LIKE ‘%A%A%’;*

OUTPUT:

ENAME
ADAMS

Accepting Values at Runtime

To create an interactive SQL command, define variables in the SQL commands. This allows the user to supply values at runtime, further enhancing the ability to reuse your scripts. SQL*Plus lets you define variables in your scripts. An ampersand (&) followed by a variable name prompts for and accepts values at runtime. For example, look at the following SELECT statement that queries the EMP table based on the department number supplied at runtime:

SQL> *SELECT empno, ename FROM emp WHERE deptno = &detpt;*

OUTPUT:

Enter value for dept: 10

old 3: WHERE deptno = &dept

new 3: WHERE deptno = 10

EMPNO	ENAME
7782	CLARK
7839	KING
7934	MILLER

3 rows selected.

** The old line with the variable and the new line with the substitution are displayed. You can turn off this display by using a SET command: SET VERIFY OFF.