Lab 2 - Queries, Arithmetic Expressions and Functions

1. Objective

This set of laboratory exercises illustrates some more advanced features of SQL and SQL*Plus expressions. The number of examples to try is large, yet you are advised to complete all of them, both in and outside your timetabled lab classes.

As with the previous handout, the main source of most of the exercises in this handout is the SQL*Plus User's Manual published by the ORACLE Corporation.

In order to test all the suggested examples, you need to have the three tables existing on your Oracle database account, namely EMP, DEPT and SALGRADE. We now begin the tour of showing more advanced features of SQL*Plus commands first.

2. SQL*PLUS LINE EDITING COMMANDS

When you type a SQL command, Oracle stores it in an area called the *SQL command buffer*. The SQL command stays in the buffer until it is replaced by the next set of SQL commands that you enter. The command in the buffer is called the "current" SQL command. SQL*Plus provides some line editing commands to help you modify the buffer:

SQL*Plus Commands

COMMAND	ABBEVIATION	PURPOSE
APPEND	A text	Add text at the end of a line
CHANGE/old/new	C/old/new	Change old to new in the line
CHANGE/text	C text	Delete text from a line
CLEAR BUFFER	CL BUFF	Delete all lines
DEL	(none)	Delete a line
INPUT	I	Add one or more lines
LIST	L	List all lines in the SQL buffer
LIST n	L n OR n	List one line
LIST LAST	L LAST	List the last line
LIST m n	L m n	List a range of lines (m to n)

2.1 Change the Current SQL Command Without Retyping It

Oracle allows you to modify the current SQL command without having to retype the entire command.

- Enter a command with the column named DEPTNO mistakenly entered as DPTNO.

```
SQL > SELECT DPTNO, ENAME, SAL
  2  FROM EMP
  3  WHERE DEPTNO = 10;
```

Oracle responds to the typing mistake with an error message. Rather than retyping the entire command you can correct the mistake by modifying the command in the buffer. Firstly you must "position" to the line of the command that contains the error by using the LIST command (abbreviated L).

- Position to line 1 (L1) of the current query

```
SQL > L1
```

Once you have positioned to the line that contains the error you can fix the mistake using the CHANGE command.

- Change DPTNO to DEPTNO

SQL > CHANGE/DPTNO/DEPTNO

The CHANGE command changes the contents of a line in the SQL command buffer and displays the new contents of the line so that you can verify the change. (Note that the CHANGE command can be abbreviated as C).

2.2 Reissuing the Current SQL Command

You may now issue the RUN command to execute the correct SQL command in the SQL command buffer.

- Execute the current SQL command

SQL > RUN

Notice that as a result of the RUN command Oracle will display the contents of the SQL command buffer and then execute the command.

2.3 Listing the Current SQL Command

If you just want to look at but not execute the current SQL command you use the LIST command

- List the contents of the SQL command buffer.

SQL > LIST

Notice the asterisk after the number 3 (3* WHERE DEPTNO = 10). The asterisk indicates that you are "positioned" to line 3 in the SQL command buffer. If you issue a CHANGE command it will be applied to line number 3.

2.4 Adding One or More Lines

In addition to being able to change a line in the buffer, you can use the INPUT command (abbreviated as "I") to add a new line after the line you are positioning to.

- Add a new line to the current SQL query.

```
SQL > I
      4 ORDER BY SAL DESC
      5
SQL > R
```

Notice that when you type I you get line number prompt (4 in the example above). You can then enter one or more new lines into the SQL command buffer. To get back to the SQL*Plus prompt you must enter a blank line. After you get back to the SQL*Plus prompt you can issue the Run command (abbreviate as "R").

2.5 The Difference Between SQL Commands and SQL*Plus Commands

CHANGE, RUN and LIST are SQL*Plus commands – ***not*** SQL Commands. SQL commands like SELECT are used to access data in tables while SQL*Plus commands like CHANGE are used to modify or "edit" the contents of the SQL command buffer. There are additional SQL*Plus command or controlling the appearance of query output that we will discuss later.

3. MORE ON THE SELECT CLAUSE

3.1 Selecting Rows Within A Certain Range

The BETWEEN operator lets you select rows that contain values within a range

- Find all the employees who earn between $1,200 to $1,400.

SQL > SELECT ENAME, JOB, SAL
2 FROM EMP
3 WHERE SAL BETWEEN 1200 AND 1400;

- An equivalent WHERE clause have been specified as: where SAL >= 1200 and SAL <=1400. The BETWEEN operator can be modified with the work NOT.

SQL > SELECT ENAME, JOB, SAL
2 FROM EMP
3 WHERE SAL NOT BETWEEN 1200 AND 1400.

NOT BETWEEN means that only rows that are outside the range will be selected.

3.2 Selecting Rows That Match A Value In A List

The IN operator lets you select rows that contain a value that match one of the values in a list of values that you supply.

- Find the employees who are clerks, analysts or salesmen.

SQL > SELECT ENAME, JOB, DEPTNO
2 FROM EMP
3 WHERE JOB IN ('CLERK', 'ANALYST', 'SALESMAN');

The IN clause can be modified with NOT to select only the rows that have a value that is NOT in the list.

- Find the employees who are NOT clerks, analysts or salesmen.

SQL > SELECT ENAME, JOB, DEPTNO
2 FROM EMP
3. WHERE JOB NOT IN ('CLERK', 'ANALYST', 'SALESMAN');

3.3 Selecting Rows That Match A Character Pattern

In addition to being able to select rows that *completely* match a value that you supply (e.g., ENAME = 'ALLEN') you can select rows that *partially* match a combination of characters that you supply.

- Find all the employees whose names begin with the letter M.

SQL > SELECT ENAME, JOB, DEPTNO
2 FROM EMP
3 WHERE ENAME LIKE 'M%';

- Find all the employees whose names end with the letter N.

SQL > SELECT ENAME, JOB, DEPTNO
2 FROM EMP
3 WHERE ENAME LIKE '%N';

The LIKE operator uses two special characters:

%	The percent character represents any string of zero or more characters
_	The underscore character means a single character position

In the example above, the letter N is to be selected regardless of how many characters precede the N. In the example below, the row will be selected only if the letter N is preceded by exactly four letters.

- Find all the employees whose name are 5 characters long and end with the letter N.

```
SQL > SELECT ENAME, JOB, DEPTNO
  2  FROM EMP
  3  WHERE ENAME LIKE '_ _ _ _ N';
```

The % and the _ can be used in combination to select the desired "pattern" of characters.

- Find all the employees whose names are not 5 characters long.

```
SQL > SELECT ENAME, JOB, DEPTNO
      2 FROM EMP
      3 WHERE ENAME NOT LIKE '_ _ _ _ _';
```

4. ORDERING ROWS OF A QUERY RESULT

In all the examples so far, the rows of a query result have been displayed in an order determined by Oracle. You can control the order in which the selected rows are to be displayed by adding an ORDER BY clause to your SELECT command.

4.1 Ordering Rows In Ascending Order

An ORDER BY clause is always the last clause in a SELECT command.

- List the employees in department 30 in order by their salary.

```
SQL > SELECT SAL, JOB, ENAME
      2 FROM EMP
      3 WHERE DEPTNO = 30
      4 ORDER BY SAL;
```

The ORDER BY clause caused a "sort" of the rows into ascending (smallest salary first) order.

4.2 Ordering Rows In Descending Order

You may also order rows into descending (largest salary first) order.

- List all the employees in DESCending order by salary.

```
SQL > SELECT SAL, JOB, ENAME,
      2 FROM EMP
      3 WHERE DEPTNO = 30
      4 ORDER BY SAL DESC;
```

Descending order is indicated by placing a DESC after the column name that you are ordering by. But ordering is not limited to sorting on a single column.

4.3 Ordering Rows Using Multiple Columns.

- Order all employees by job, and within job put them in descending salary order.

```
SQL > SELECT JOB, SAL, ENAME
      2 FROM EMP
      3 ORDER BY JOB, SAL DESC;
```

CLERKs are in descending salary order, and all the MANAGERs are in descending salary order, etc.

4.4 Ordering Rows By Columns With NULL Values

When you order by columns with null values, Oracle 9i will consider the null entries as the largest number and put them into the end of the list if ascending. Or you can tell the DBMS the way you want to sort the NULL values. The following code segment indicates this:

SELECT ... ORDER BY ... ASC/DESC NULLS FIRST/LAST;

- Order the employees in department 30 in ascending order of their commission.

SQL > SELECT COMM, SAL, ENAME
2 FROM EMP
3 WHERE DEPTNO = 30
4 ORDER BY COMM;

- Order the employees in department 30 in descending order of the commission.

SQL > SELECT COMM, SAL, ENAME
2 FROM EMP
3 WHERE DEPTNO = 30
4 ORDER BY COMM DESC;

5. EXPRESSION AND FUNCTIONS

5.1 Arithmetic Expressions

A SQL command can contain arithmetic expressions. An arithmetic expression is made up of column names and constant numeric values connected by arithmetic operators. Any one of the following arithmetic operators may be used:

Symbol	Operation	Symbol	Operation
+	Addition	-	Subtraction
*	Multiplication	/	Division

An arithmetic expression can be used in a SELECT clause to retrieve calculated results.

- List the name, salary, commission, and sum of salary plus commission of all salesmen.

SQL > SELECT ENAME, SAL, COMM, SAL+COMM
2 FROM EMP
3 WHERE JOB = 'SALESMAN';

An arithmetic expression can be used in a WHERE clause to select rows based upon a calculated search-condition.

- List the name, salary and commission of employees whose commission is greater than 25% of their salary.

SQL > SELECT ENAME, SAL, COMM
2 FROM EMP
3 WHERE COMM > 0.25 * SAL;

Arithmetic-expression can also be used in an ORDER BY clause.

- List all salesmen in descending order of their commission divided by their salary.

SQL > SELECT ENAME, COMM/SAL, COMM, SAL
2 FROM EMP
3 WHERE JOB = 'SALESMAN'
4 ORDER BY COMM/SAL DESC;

More than one arithmetic operator can be used in an arithmetic expression. Parentheses are used to control the order of the operations to be performed.

- Calculate the total annual compensation of all salesmen based upon their monthly salary and their monthly commission.

```
SQL > SELECT ENAME, SAL, COMM, (SAL+COMM)*12
      2 FROM EMP
      3 WHERE JOB = 'SALESMAN';
```

5.2 The NULL Function

When an expression or function references a column that contains a null value, the result is null.

- Select SAL+COMM where some of the employees have a null commission.

```
SQL > SELECT ENAME, JOB, SAL, COMM, SAL+COMM
      2 FROM EMP
      3 WHERE DEPTNO = 30;
```

The expression SAL+COMM returns a null value for all employees that have a null value in their COMM field. The Oracle NULL-values, NVL, can be used to assign a temporary value to nulls encountered within an expression.

- Treat null commissions as zero commissions within the expression SAL+COMM.

```
SQL > SELECT ENAME, JOB, SAL, COMM, SAL+NVL(COMM, 0)
      2 FROM EMP
      3 WHERE DEPTNO = 30;
```

The expression SAL+NVL(COMM, 0) will return a value equal to SAL + 0 when the COMM field is null.

5.3 Column Labels

Column names are normally used as display headings for query results. To make query results more readable you can use column labels to "rename" either column names or arithmetic expressions in the SELECT clause. The column label will then be used as the display heading in place of either the column name or expression.

- Issue the previous query with the column label SALARY replacing SAL, COMMISSION replacing COMM and TOTAL_COMPENSATION replacing the expression SAL+NVL(COMM, 0).

```
SQL > SELECT ENAME, JOB, SAL SALARY, COMM COMMISSION,
      2 SAL+NVL(COMM, 0) TOTAL_COMPENSATION
      3 FROM EMP
      4 WHERE DEPTNO = 30;
```

As shown, column labels follow column names or expressions separated by a blank. Notice the use of the underscore (_) between TOTAL and COMPENSATION in the example above. This is because column labels, table names, column names, etc. may not contain embedded blanks or special characters (+,-,*,/) without being enclosing in double quotes (").

- Issue the previous query with the column label containing an embedded blank.

```
SQL > SELECT ENAME, JOB, SAL SALARY, COMM COMMISSION,
      2  SAL+NVL(COMM, 0) "TOTAL COMPENSATION"
      3  FROM EMP
```

```
4 WHERE DEPTNO = 30;
```

5.4 Arithmetic Functions

In addition to arithmetic operators (+,-,*,/), you can have *arithmetic functions* in a SQL command. The arithmetic functions include:

ROUND(number[, d])	Rounds number to d digits right of the decimal point
TRUNC (number[, d])	Truncates number to d digits right of the decimal point
ABS (number)	Absolute value of the number
SIGN (number)	+1 if number > 0; 0 if number = 0; -1 if number < 0
TO_CHAR (number)	Converts a number to a character
MOD (num1, num2)	Num1 modulo num2

When an arithmetic function is used, the column name must be enclosed in parentheses.

- Calculate the daily salary of the department 30 employees. Select the results in three separate columns: unrounded, rounded to the nearest dollar, and rounded to the nearest cent. For the purposes of the calculation assume there are 22 working days in a month.

```
SQL > SELECT ENAME, SAL, SAL/22, ROUND(SAL/22, 0), ROUND(SAL/22, 2)
  2 FROM EMP
  3 WHERE DEPTNO = 30;
```

In the example above, the first expression (SAL/22) was not rounded at all. The second expression (round(SAL/22,0)) was rounded to the nearest dollar. The third expression (round(SAL/22,2)) was rounded to the nearest cent (tow digits to the right of the decimal point).

- Compute the daily and hourly salary for employees in department 30. Round the results to the nearest cent. Assume there are 22 working days in a month and 8 working hours in a day.

```
SQL > SELECT ENAME, SAL MONTHLY, ROUND(SAL/22,2) DAILY,
  2 ROUND(SAL/(22*8),2) HOURLY
  3 FROM EMP
  4 WHERE DEPTNO = 30;
```

- Issue the same query as the previous one except this time truncate (TRUNC) to the nearest cent rather than round.

```
SQL > SELECT ENAME, SAL MONTHLY, TRUNC(SAL/22,2) DAILY,
  2 TRUNC(SAL/(22*8),2) HOURLY
  3 FROM EMP
  4 WHERE DEPTNO = 30;
```

Notice the difference between the results of the previous query that used truncation, and the one before it that used rounding.

5.5 Character String Functions

In addition to functions that operate on numbers (arithmetic functions), Oracle provides you with functions that operate on character data. These functions are called *character string functions*. (Note that some character string functions allow you to

operate on a combination of character fields and numbers.) The character string functions include:

string1 || string2	Concatenates string1 with string2
LENGTH(string)	Length of a string
SUBSTR(str,spos[,len]	Substring of a string
INSTR(sstr,str[,spos])	Position of substring in the string
UPPER(string)	Changes all lower case characters to upper case
LOWER(string)	Changes all upper case characters to lower case
TO_NUM(string)	Converts a character to a number
LPAD(string,len[,chr])	Left pads the string with fill character
RPAD(string,len[,chr])	Right pads the string with fill character

The *concatenation function* allows you to combine several columns and/or constant values into a single column.

- Concatenate the DNAME column to the LOC column separated by a blank space, a dash and another blank space (‘ – ‘).

SQL > SELECT DNAME || ‘ – ‘ || LOC DEPARTMENTS
2 FROM DEPT;

5.6 Substring

- Abbreviate the department name to the first 5 characters of the department name.

SQL > SELECT SUBSTR(DNAME,1,5)
2 FROM DEPT;

- Select an employee name in both UPPER and LOWER case.

SQL > SELECT ENAME, UPPER(ENAME),LOWER(ENAME)
2 FROM EMP
3 WHERE UPPER(ENAME) = UPPER(‘Ward’);

Note the use of the UPPER function on the WHERE clause to insure that the case would match.

- Find the number of characters in department names.

SQL > SELECT DNAME, LENGTH(DNAME)
2 FROM DEPT;