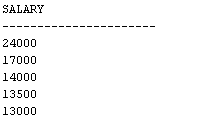
**Practice 7**

1. Create a PL/SQL block that determines the top *n* salaries of the employees.

1. Execute the lab\_07\_01.sql script to create a new table, top\_salaries, for storing the salaries of the employees.

create table top\_salaries(salary number(10));

1. In the declarative section, declare a variable v\_num of type NUMBER that holds a number n representing the number of top *n* earners from the employees table. For example, to view the top five salaries, enter 5. Declare another variable sal of type employees.salary. Declare a cursor, c\_emp\_cursor, that retrieves the salaries of employees in descending order.
2. In the executable section, open the loop and fetch top *n* salaries and insert them into top\_salaries table. You can use a simple loop to operate on the data. Also, try and use %ROWCOUNT and %FOUND attributes for the exit condition.
3. After inserting into the top\_salaries table, display the rows with a SELECT statement. The output shown represents the five highest salaries in the employees table.



declare

v\_num number;

sal employees.salary%type;

Cursor c\_emp\_cursor is

select distinct salary from employees order by

salary desc;

begin

open c\_emp\_cursor;

loop

fetch c\_emp\_cursor into sal;

exit when c\_emp\_cursor%ROWCOUNT>&v\_num or not c\_emp\_cursor%FOUND;

insert into top\_salaries values(sal);

end loop;

close c\_emp\_cursor;

end;

1. Test a variety of special cases, such as v\_num = 0 or where v\_num is greater than the number of employees in the employees table. Empty the top\_salaries table after each test.

2. Create a PL/SQL block that does the following:

1. In the declarative section, declare a variable v\_deptno of type NUMBER and assign a value that holds department ID.
2. Declare a cursor, c\_emp\_cursor, that retrieves the last\_name, salary, and manager\_id of the employees working in the department specified in v\_deptno.
3. In the executable section, use the cursor FOR loop to operate on the data retrieved. If the salary of the employee is less than 5,000 and if the manager ID is either 101 or 124, display the message <<*last\_name*>> Due for a raise. Otherwise, display the message <<*last\_name*>> Not due for a raise.
4. Test the PL/SQL block for the following cases:



set serveroutput on;

declare

v\_sal number(10);

v\_deptno number ;

cursor c\_emp\_cursor is

select last\_name, salary, manager\_id,department\_id from employees;

c\_emp\_record c\_emp\_cursor%rowtype;

begin

for c\_emp\_record in c\_emp\_cursor

loop

if c\_emp\_record.salary < 5000 and c\_emp\_record.manager\_id in (101,124)

then dbms\_output.put\_line(c\_emp\_record.department\_id||c\_emp\_record.last\_name ||' '|| 'due for raise');

else

dbms\_output.put\_line(c\_emp\_record.department\_id||c\_emp\_record.last\_name ||' '|| ' not due for raise');

end if;

end loop;

end;

1. Write a PL/SQL block that declares and uses cursors with parameters.  
   In a loop, use a cursor to retrieve the department number and the department name from the departments table for a department whose department\_id is less than 100. Pass the department number to another cursor as a parameter to retrieve from the employees table the details of employee last name, job, hire date, and salary of those employees whose employee\_id is less than 120 and who work in that department.
2. In the declarative section, declare a cursor dept\_cursor to retrieve department\_id and department\_name for those departments with department\_id less than 100. Order by department\_id.
3. Declare another cursor emp\_cursor that takes the department number as parameter and retrieves last\_name, job\_id, hire\_date, and salary of those employees whose employee\_id is less than 120 and who work in that department.
4. Declare variables to hold the values retrieved from each cursor. Use the %TYPE attribute while declaring variables.
5. Open the dept\_cursor, use a simple loop, and fetch values into the variables declared. Display the department number and department name.
6. For each department, open emp\_cursor by passing the current department number as a parameter. Start another loop and fetch the values of emp\_cursor into variables and print all the details retrieved from the employees table.  
   **Note:** You may want to print a lineafter you have displayed the details of each department.Use appropriate attributes for the exit condition. Also, determine whether a cursor is already open before opening the cursor.
7. Close all the loops and cursors, and then end the executable section. Execute the script.

DEFINE dept\_id\_constraint = 100;

DEFINE emp\_id\_constraint = 120;

DECLARE

CURSOR c\_lower\_depts

(higher\_id NUMBER)

IS

SELECT department\_id, department\_name

FROM departments

WHERE department\_id < higher\_id;

CURSOR c\_lower\_emps\_in\_dept

(higher\_emp\_id NUMBER, dept\_id NUMBER)

IS

SELECT e.last\_name, e.job\_id, j.job\_title, e.hire\_date, e.salary

FROM employees e, jobs j

WHERE

e.employee\_id < higher\_emp\_id AND

e.department\_id = dept\_id AND

e.job\_id = j.job\_id;

BEGIN

FOR v\_dept IN c\_lower\_depts(&dept\_id\_constraint) LOOP

DBMS\_OUTPUT.PUT\_LINE('----------');

DBMS\_OUTPUT.PUT(CHR(10)); -- output a newline character

DBMS\_OUTPUT.PUT('Department Number: ' || v\_dept.department\_id);

DBMS\_OUTPUT.PUT\_LINE('; Department Name: ' || v\_dept.department\_name);

DBMS\_OUTPUT.PUT(CHR(10));

FOR v\_emp IN

c\_lower\_emps\_in\_dept(&emp\_id\_constraint, v\_dept.department\_id)

LOOP

DBMS\_OUTPUT.PUT(v\_emp.last\_name || ' ');

DBMS\_OUTPUT.PUT(v\_emp.job\_title || ' (' || v\_emp.job\_id || ') ');

DBMS\_OUTPUT.PUT\_LINE(v\_emp.hire\_date || ' ' || v\_emp.salary);

END LOOP;

DBMS\_OUTPUT.PUT(CHR(10));

END LOOP;

END;

/

