PL/SQL_06

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

   a) In the declarative section, declare a variable **v_deptno** of type NUMBER. Assign a valid department ID value (see table in step d for values).

   b) 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**.

   c) 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.

   d) Test the PL/SQL block for the following cases:

Department ID	**Message**
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise . . . . Oconnel Due for a raise Grant Due for a raise
80	Russell Not Due for a raise Partners Not Due for a raise Errazuriz Not Due for a raise Cambrault Not Due for a raise . . . . Livingston Not Due for a raise Johnson Not Due for a raise

2. 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.

   a) 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.

   b) 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.

   c) Declare variables to hold the values retrieved from each cursor. Use the %TYPE attribute while declaring variables.

   d) Open the dept_cursor, use a simple loop, and fetch values into the variables declared. Display the department number and department name.

   e) 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 line after 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.

   f) Close all the loops and cursors, and then end the executable section.

   g) Execute the script. The sample output is as follows:

```
Department number 10 Department name Administration
-----------------------------------------------------------
Department number 20 Department name Marketing
-----------------------------------------------------------
Department number 30 Department name Purchasing
Raphaely PU_MAN 2002.12.07 11000
Khoo PU_CLERK 2003.05.18 3100
Baida PU_CLERK 2005.12.24 2900
Tobias PU_CLERK 2005.07.24 2800
Himuro PU_CLERK 2006.11.15 2600
Colmenares PU_CLERK 2007.08.10 2500
-----------------------------------------------------------
Department number 40 Department name Human Resources
-----------------------------------------------------------
Department number 50 Department name Shipping
-----------------------------------------------------------
Department number 60 Department name IT
Hunold IT_PROG 2006.01.03 9000
Ernst IT_PROG 2007.05.21 6000
Austin IT_PROG 2005.06.25 4800
Pataballa IT_PROG 2006.02.05 4800
Lorentz IT_PROG 2007.02.07 4200
-----------------------------------------------------------
Department number 70 Department name Public Relations
-----------------------------------------------------------
Department number 80 Department name Sales
-----------------------------------------------------------
Department number 90 Department name Executive
King AD_PRES 2003.06.17 24000
Kochhar AD_VP 2005.09.21 17000
De Haan AD_VP 2001.01.13 17000
-----------------------------------------------------------
```

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

   a) Execute the lab_07_01.sql script to create a new table, top_salaries, for storing the salaries of the employees:

```
-- lab_07_01.sql

DROP TABLE top_salaries;
/
CREATE TABLE top_salaries (salary           NUMBER(8,2));
/
```

   b) 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.

   c) 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.
   Note: Make sure you add an exit condition to avoid having an infinite loop.

   d) 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:

```
SALARY
----------
   24000
   17000
   17000
   14000
   13500
```

   e) 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.