

Practices for Lesson 8: Using Explicit Cursors

Chapter 8

Practice 8-1: Using Explicit Cursors

In this practice, you perform two exercises:

- First, you use an explicit cursor to process a number of rows from a table and populate another table with the results by using a cursor `FOR` loop.
 - Second, you write a PL/SQL block that processes information with two cursors, including one that uses a parameter.
1. Create a PL/SQL block to perform the following:
 - a. In the declarative section, declare and initialize a variable named `v_deptno` of type `NUMBER`. Assign a valid department ID value (see table in step d for values).
 - b. Declare a cursor named `c_emp_cursor`, which retrieves the `last_name`, `salary`, and `manager_id` of 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:

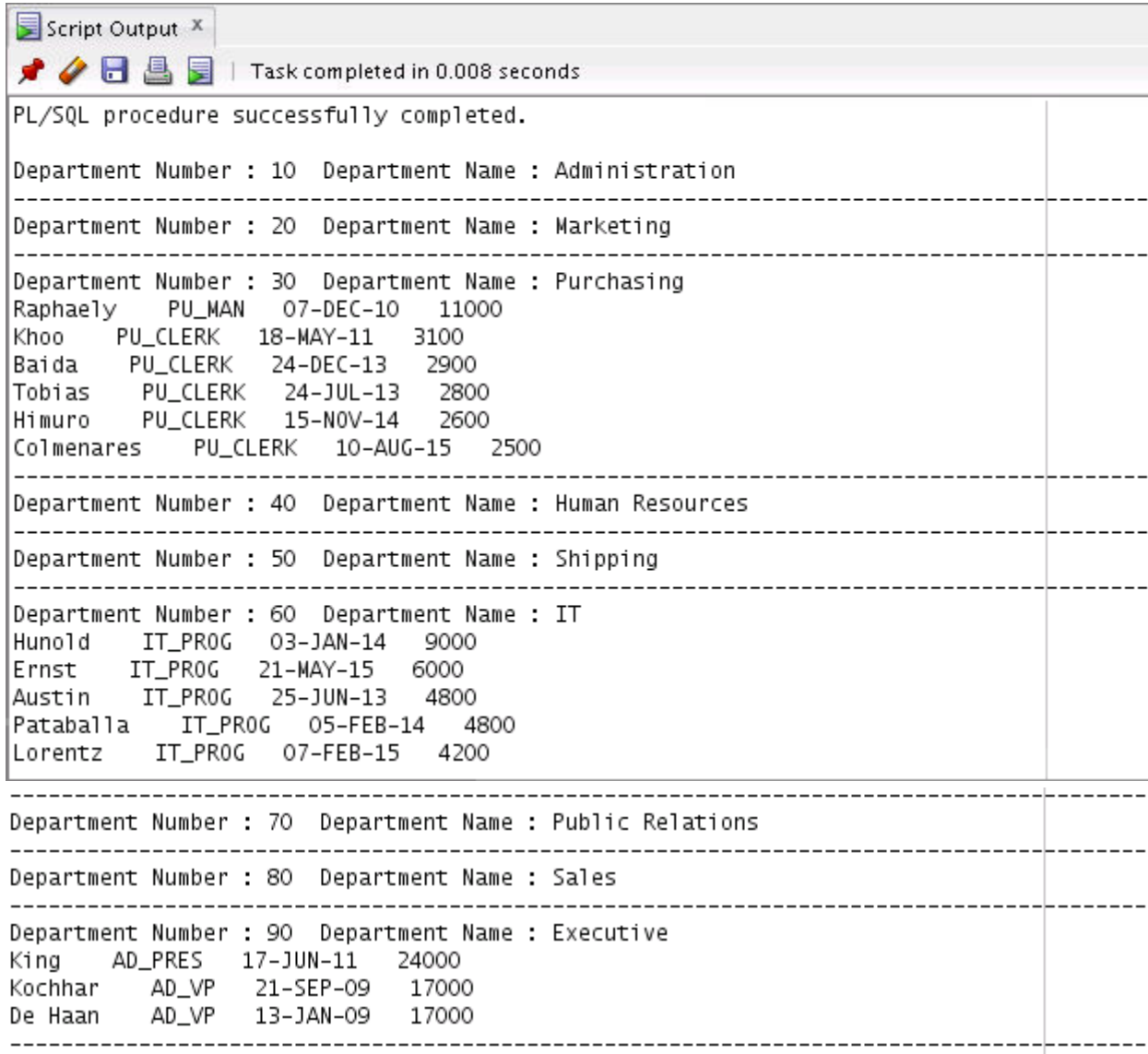
Department ID	Message
10	Whalen Due for a raise
20	Hartstein Not Due for a raise Fay Not Due for a raise
50	Weiss Not Due for a raise Fripp Not Due for a raise Kaufling Not Due for a raise Vollman Not Due for a raise. OConnell Due for a raise Grant Due for a raise
80	Russell Not Due for a raise Partners 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. Next, write a PL/SQL block that declares and uses two cursors—one without a parameter and one with a parameter. The first cursor retrieves the department number and department name from the `DEPARTMENTS` table for all departments whose ID number is less than 100. The second cursor receives the department number as a parameter, and retrieves employee details for those who work in that department and whose `employee_id` is less than 120.
- Declare a cursor `c_dept_cursor` to retrieve `department_id` and `department_name` for those departments with `department_id` less than 100. Order by `department_id`.
 - Declare another cursor `c_emp_cursor` that takes the department number as parameter and retrieves the following data from the `EMPLOYEES` table: `last_name`, `job_id`, `hire_date`, and `salary` of those employees who work in that department, with `employee_id` less than 120.
 - Declare variables to hold the values retrieved from each cursor. Use the `%TYPE` attribute while declaring variables.
 - Open `c_dept_cursor` and use a simple loop to fetch values into the variables that are declared. Display the department number and department name. Use the appropriate cursor attribute to exit the loop.
 - Open `c_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.

Notes

- Check whether `c_emp_cursor` is already open before opening the cursor.
 - Use the appropriate cursor attribute for the exit condition.
 - When the loop completes, print a line after you have displayed the details of each department, and close `c_emp_cursor`.
- f. End the first loop and close `c_dept_cursor`. Then end the executable section.

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



```
PL/SQL procedure successfully completed.

Department Number : 10  Department Name : Administration
-----
Department Number : 20  Department Name : Marketing
-----
Department Number : 30  Department Name : Purchasing
Raphaely      PU_MAN      07-DEC-10      11000
Khoo          PU_CLERK     18-MAY-11      3100
Baida         PU_CLERK     24-DEC-13      2900
Tobias        PU_CLERK     24-JUL-13      2800
Himuro        PU_CLERK     15-NOV-14      2600
Colmenares    PU_CLERK     10-AUG-15      2500
-----
Department Number : 40  Department Name : Human Resources
-----
Department Number : 50  Department Name : Shipping
-----
Department Number : 60  Department Name : IT
Hunold        IT_PROG      03-JAN-14      9000
Ernst         IT_PROG      21-MAY-15      6000
Austin        IT_PROG      25-JUN-13      4800
Pataballa     IT_PROG      05-FEB-14      4800
Lorentz       IT_PROG      07-FEB-15      4200
-----
Department Number : 70  Department Name : Public Relations
-----
Department Number : 80  Department Name : Sales
-----
Department Number : 90  Department Name : Executive
King          AD_PRES      17-JUN-11      24000
Kochhar       AD_VP        21-SEP-09      17000
De Haan       AD_VP        13-JAN-09      17000
-----
```

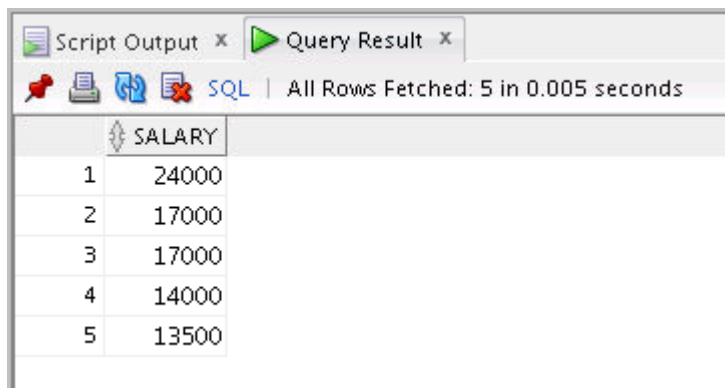
Practice 8-2: Using Explicit Cursors: Optional

If you have time, complete the following optional practice. Here, create a PL/SQL block that uses an explicit cursor to determine the top *n* salaries of employees.

1. Run the `lab_08-02.sql` script to create the `TOP_SALARIES` table for storing the salaries of the employees.
2. In the declarative section, declare the `v_num` variable of the `NUMBER` type 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 `v_sal` of type `employees.salary`. Declare a cursor, `c_emp_cursor`, which retrieves the salaries of employees in descending order. Remember that the salaries should not be duplicated.
3. In the executable section, open the loop, fetch the top *n* salaries, and then insert them into the `TOP_SALARIES` table. You can use a simple loop to operate on the data. Also, try and use the `%ROWCOUNT` and `%FOUND` attributes for the exit condition.

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

4. After inserting data into the `TOP_SALARIES` table, display the rows with a `SELECT` statement. The output shown represents the five highest salaries in the `EMPLOYEES` table.



The screenshot shows a SQL query result window with a tab labeled 'Query Result'. Below the tab, there are icons for saving, printing, and refreshing, followed by the text 'SQL | All Rows Fetched: 5 in 0.005 seconds'. The main area displays a table with two columns: an index from 1 to 5 and a column labeled 'SALARY'. The salary values are 24000, 17000, 17000, 14000, and 13500 respectively.

	SALARY
1	24000
2	17000
3	17000
4	14000
5	13500

5. 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.