Practices for Lesson 7: Working with Composite Data Types

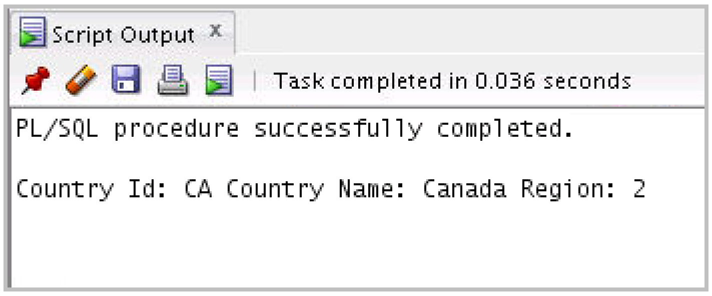
Chapter 7

Practice 7: Working with Composite Data Types

Note: If you have executed the code examples for this lesson, make sure that you execute the following code before starting this practice:

```
DROP table retired_emps;
DROP table empl;
```

1. Write a PL/SQL block to print information about a given country.
   a. Declare a PL/SQL record based on the structure of the `COUNTRIES` table.
   b. Declare a variable `v_countryid`. Assign `CA` to `v_countryid`.
   c. In the declarative section, use the `%ROWTYPE` attribute and declare the `v_country_record` variable of type `countries`.
   d. In the executable section, get all the information from the `COUNTRIES` table by using `v_countryid`. Display selected information about the country. The sample output is as follows:



```
PL/SQL procedure successfully completed.

Country Id: CA Country Name: Canada Region: 2
```

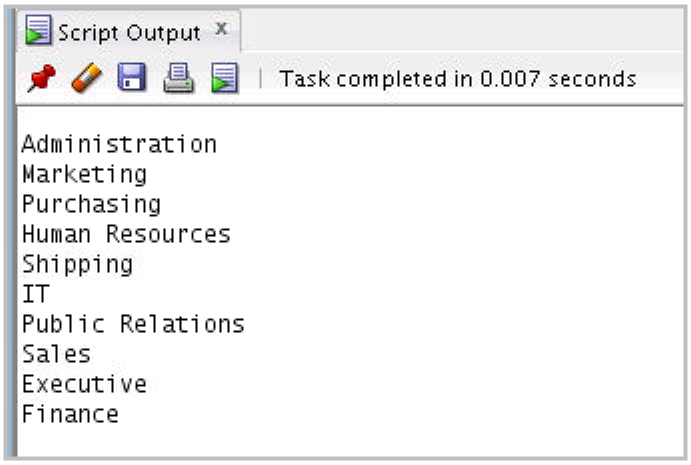
   e. You may want to execute and test the PL/SQL block for countries with the IDs DE, UK, and US.
2. Create a PL/SQL block to retrieve the names of some departments from the `DEPARTMENTS` table and print each department name on the screen, incorporating an associative array. Save the script as `lab_07_02_soln.sql`.
   a. Declare an `INDEX BY` table `dept_table_type` of type `departments.department_name`. Declare a variable `my_dept_table` of type `dept_table_type` to temporarily store the names of the departments.
   b. Declare two variables: `f_loop_count` and `v_deptno` of type `NUMBER`. Assign 10 to `f_loop_count` and 0 to `v_deptno`.
   c. Using a loop, retrieve the names of 10 departments and store the names in the associative array. Start with `department_id` 10. Increase `v_deptno` by 10 for every loop iteration. The following table shows the `department_id` for which you should retrieve the `department_name`.

DEPARTMENT_ID	DEPARTMENT_NAME
10	**Administration**
20	**Marketing**
30	**Purchasing**
40	**Human Resources**

50	**Shipping**
60	**IT**
70	**Public Relations**
80	**Sales**
90	**Executive**
100	**Finance**

d. Using another loop, retrieve the department names from the associative array and display them.

e. Execute and save your script as `lab_07_02_soln.sql`. The output is as follows:



```
Administration
Marketing
Purchasing
Human Resources
Shipping
IT
Public Relations
Sales
Executive
Finance
```

3. Modify the block that you created in Task 2 to retrieve all information about each department from the `DEPARTMENTS` table and display the information. Use an associative array with the `INDEX BY` table of records method.

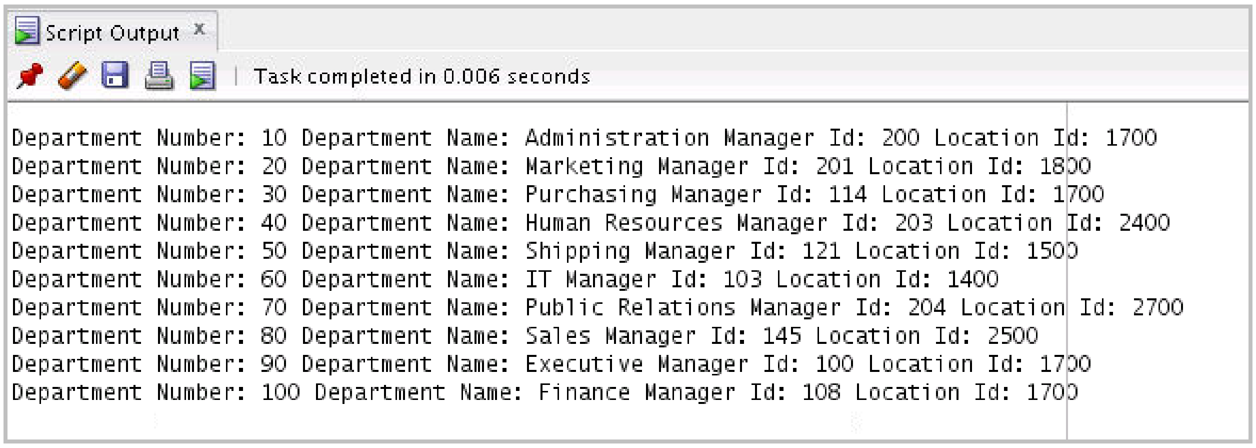
   a. Load the `lab_07_02_soln.sql` script.

   b. You have declared the associative array to be of type `departments.department_name`. Modify the declaration of the associative array to temporarily store the number, name, and location of all the departments. Use the `%ROWTYPE` attribute.

   c. Modify the `SELECT` statement to retrieve all department information currently in the `DEPARTMENTS` table and store it in the associative array.

   d. Using another loop, retrieve the department information from the associative array and display the information.

The sample output is as follows:



```
Department Number: 10 Department Name: Administration Manager Id: 200 Location Id: 1700
Department Number: 20 Department Name: Marketing Manager Id: 201 Location Id: 1800
Department Number: 30 Department Name: Purchasing Manager Id: 114 Location Id: 1700
Department Number: 40 Department Name: Human Resources Manager Id: 203 Location Id: 2400
Department Number: 50 Department Name: Shipping Manager Id: 121 Location Id: 1500
Department Number: 60 Department Name: IT Manager Id: 103 Location Id: 1400
Department Number: 70 Department Name: Public Relations Manager Id: 204 Location Id: 2700
Department Number: 80 Department Name: Sales Manager Id: 145 Location Id: 2500
Department Number: 90 Department Name: Executive Manager Id: 100 Location Id: 1700
Department Number: 100 Department Name: Finance Manager Id: 108 Location Id: 1700
```

Solution 7: Working with Composite Data Types

1. Write a PL/SQL block to print information about a given country.
   a. Declare a PL/SQL record based on the structure of the COUNTRIES table.
   b. Declare a variable v_countryid. Assign CA to v_countryid.

```
SET SERVEROUTPUT ON

SET VERIFY OFF
DECLARE
  v_countryid varchar2(20):= 'CA';
```

   c. In the declarative section, use the %ROWTYPE attribute and declare the v_country_record variable of type countries.

```
 v_country_record countries%ROWTYPE;
```

   d. In the executable section, get all the information from the COUNTRIES table by using v_countryid. Display selected information about the country.

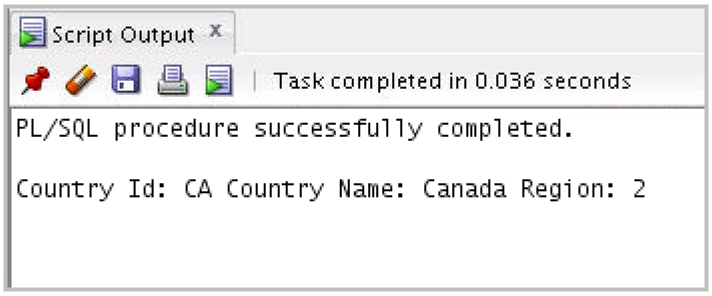
```

BEGIN
 SELECT *
 INTO   v_country_record
 FROM   countries
 WHERE country_id = UPPER(v_countryid);

 DBMS_OUTPUT.PUT_LINE ('Country Id: ' ||
     v_country_record.country_id ||
     ' Country Name: ' || v_country_record.country_name
     || ' Region: ' || v_country_record.region_id);

END;
```

The sample output after performing all the above steps is as follows:



```
PL/SQL procedure successfully completed.

Country Id: CA Country Name: Canada Region: 2
```

   e. You may want to execute and test the PL/SQL block for countries with the IDs DE, UK, and US.

2. Create a PL/SQL block to retrieve the names of some departments from the `DEPARTMENTS` table and print each department name on the screen, incorporating an associative array. Save the script as `lab_07_02_soln.sql`.

   a. Declare an `INDEX BY` table `dept_table_type` of type `departments.department_name`. Declare a variable `my_dept_table` of type `dept_table_type` to temporarily store the names of the departments.

```
SET SERVEROUTPUT ON

DECLARE
   TYPE dept_table_type is table of
     departments.department_name%TYPE
   INDEX BY PLS_INTEGER;
   my_dept_table   dept_table_type;
```

   b. Declare two variables: `f_loop_count` and `v_deptno` of type `NUMBER`. Assign 10 to `f_loop_count` and 0 to `v_deptno`.

```
   f_loop_count    NUMBER (2):=10;
   v_deptno        NUMBER (4):=0;
```

   c. Using a loop, retrieve the names of 10 departments and store the names in the associative array. Start with `department_id` 10. Increase `v_deptno` by 10 for every iteration of the loop. The following table shows the `department_id` for which you should retrieve the `department_name` and store in the associative array.

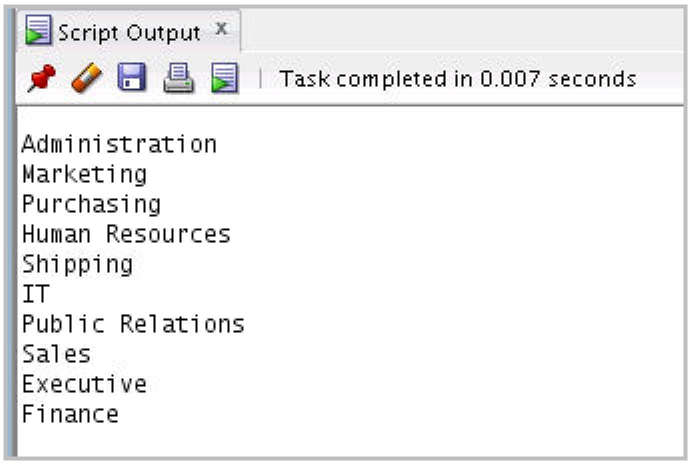
DEPARTMENT_ID	DEPARTMENT_NAME
10	**Administration**
20	**Marketing**
30	**Purchasing**
40	**Human Resources**
50	**Shipping**
60	**IT**
70	**Public Relations**
80	**Sales**
90	**Executive**
100	**Finance**

```
BEGIN
  FOR i IN 1..f_loop_count
  LOOP
        v_deptno:=v_deptno+10;
   SELECT department_name
   INTO my_dept_table(i)
   FROM departments
   WHERE department_id = v_deptno;
  END LOOP;
```

d. Using another loop, retrieve the department names from the associative array and display them.

```
FOR i IN 1..f_loop_count
  LOOP
    DBMS_OUTPUT.PUT_LINE (my_dept_table(i));
  END LOOP;
END;
```

e. Execute and save your script as `lab_07_02_soln.sql`. The output is as follows:



```
Administration
Marketing
Purchasing
Human Resources
Shipping
IT
Public Relations
Sales
Executive
Finance
```

3. Modify the block that you created in Task 2 to retrieve all information about each department from the `DEPARTMENTS` table and display the information. Use an associative array with the `INDEX BY` table of records method.
   a. Load the `lab_07_02_soln.sql` script.
   b. You have declared the associative array to be of the `departments.department_name` type. Modify the declaration of the associative array to temporarily store the number, name, and location of all the departments. Use the `%ROWTYPE` attribute.

```
SET SERVEROUTPUT ON

DECLARE
   TYPE dept_table_type is table of departments%ROWTYPE
    INDEX BY PLS_INTEGER;
    my_dept_table   dept_table_type;
    f_loop_count        NUMBER (2):=10;
    v_deptno           NUMBER (4):=0;
```

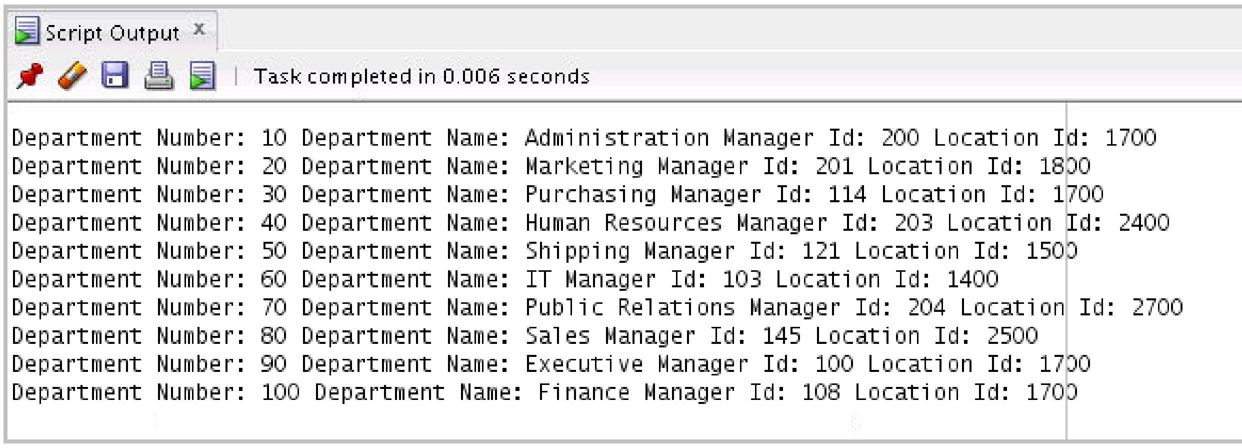
c. Modify the `SELECT` statement to retrieve all department information currently in the `DEPARTMENTS` table and store it in the associative array.

```
BEGIN
  FOR i IN 1..f_loop_count
  LOOP
   v_deptno := v_deptno + 10;
   SELECT *
   INTO my_dept_table(i)
   FROM departments
   WHERE department_id = v_deptno;
  END LOOP;
```

d. Using another loop, retrieve the department information from the associative array and display the information.

```
FOR i IN 1..f_loop_count
  LOOP
    DBMS_OUTPUT.PUT_LINE ('Department Number: ' ||
my_dept_table(i).department_id
     || ' Department Name: ' || my_dept_table(i).department_name
     || ' Manager Id: '||  my_dept_table(i).manager_id
     || ' Location Id: ' || my_dept_table(i).location_id);
  END LOOP;
END;
```

The sample output is as follows:



```
Department Number: 10 Department Name: Administration Manager Id: 200 Location Id: 1700
Department Number: 20 Department Name: Marketing Manager Id: 201 Location Id: 1800
Department Number: 30 Department Name: Purchasing Manager Id: 114 Location Id: 1700
Department Number: 40 Department Name: Human Resources Manager Id: 203 Location Id: 2400
Department Number: 50 Department Name: Shipping Manager Id: 121 Location Id: 1500
Department Number: 60 Department Name: IT Manager Id: 103 Location Id: 1400
Department Number: 70 Department Name: Public Relations Manager Id: 204 Location Id: 2700
Department Number: 80 Department Name: Sales Manager Id: 145 Location Id: 2500
Department Number: 90 Department Name: Executive Manager Id: 100 Location Id: 1700
Department Number: 100 Department Name: Finance Manager Id: 108 Location Id: 1700
```