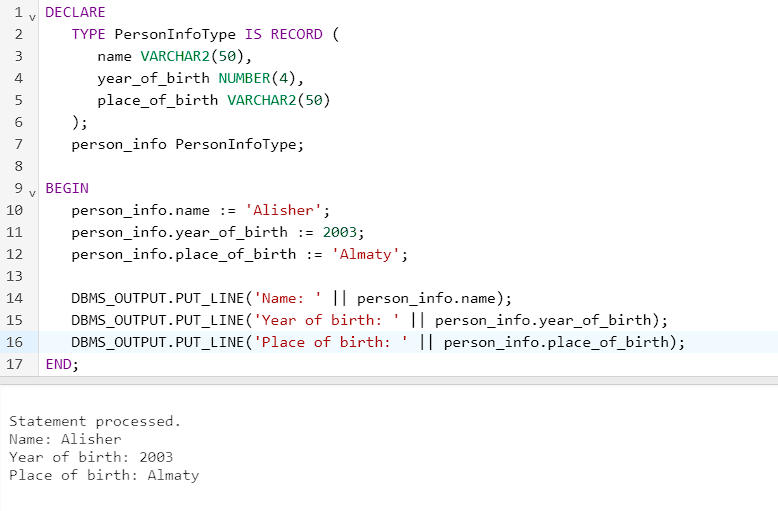
**Practice 4.1 (Composite Data Types)**

1. Within the PL/SQL block create a record to store information about yourself: name, year of birth, place of birth. Output the information to the screen.



DECLARE

TYPE PersonInfoType IS RECORD (

name VARCHAR2(50),

year\_of\_birth NUMBER(4),

place\_of\_birth VARCHAR2(50)

);

person\_info PersonInfoType;

BEGIN

person\_info.name := 'Alisher';

person\_info.year\_of\_birth := 2003;

person\_info.place\_of\_birth := 'Almaty';

DBMS\_OUTPUT.PUT\_LINE('Name: ' || person\_info.name);

DBMS\_OUTPUT.PUT\_LINE('Year of birth: ' || person\_info.year\_of\_birth);

DBMS\_OUTPUT.PUT\_LINE('Place of birth: ' || person\_info.place\_of\_birth);

END;

/

2. Write a PL/SQL block to print information about a given country.

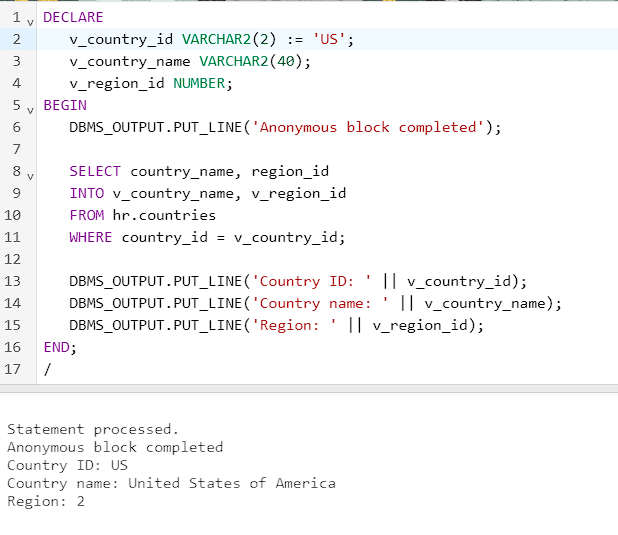
a. Declare a PL/SQL record based on the structure of the countries table.

b. Define a variable countryid and pass the value througha substitution variable.

1. In the executable section, get all the information from the countries table by using countryid. Display selected information about the country. A sample output is shown below.
2. You may want to execute and test the PL/SQL block for the countries with the IDs DE, UK, US.

Изображение выглядит как текст, Шрифт, белый, снимок экрана

Автоматически созданное описание



DECLARE

v\_country\_id VARCHAR2(2) := 'UK';

v\_country\_name VARCHAR2(40);

v\_region\_id NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Anonymous block completed');

SELECT country\_name, region\_id

INTO v\_country\_name, v\_region\_id

FROM hr.countries

WHERE country\_id = v\_country\_id;

DBMS\_OUTPUT.PUT\_LINE('Country ID: ' || v\_country\_id);

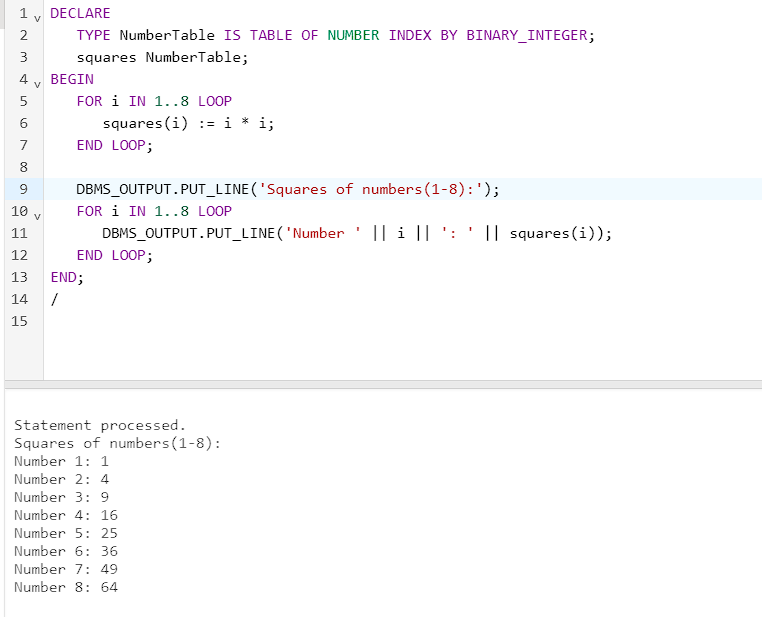
DBMS\_OUTPUT.PUT\_LINE('Country name: ' || v\_country\_name);

DBMS\_OUTPUT.PUT\_LINE('Region: ' || v\_region\_id);

END;

/

1. Within a PL/SQL block create an INDEX BY table to store the square of the numbers from 1 to 8. Output the result to the screen.



DECLARE

TYPE NumberTable IS TABLE OF NUMBER INDEX BY BINARY\_INTEGER;

squares NumberTable;

BEGIN

FOR i IN 1..8 LOOP

squares(i) := i \* i;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Squares of numbers(1-8):');

FOR i IN 1..8 LOOP

DBMS\_OUTPUT.PUT\_LINE('Number ' || i || ': ' || squares(i));

END LOOP;

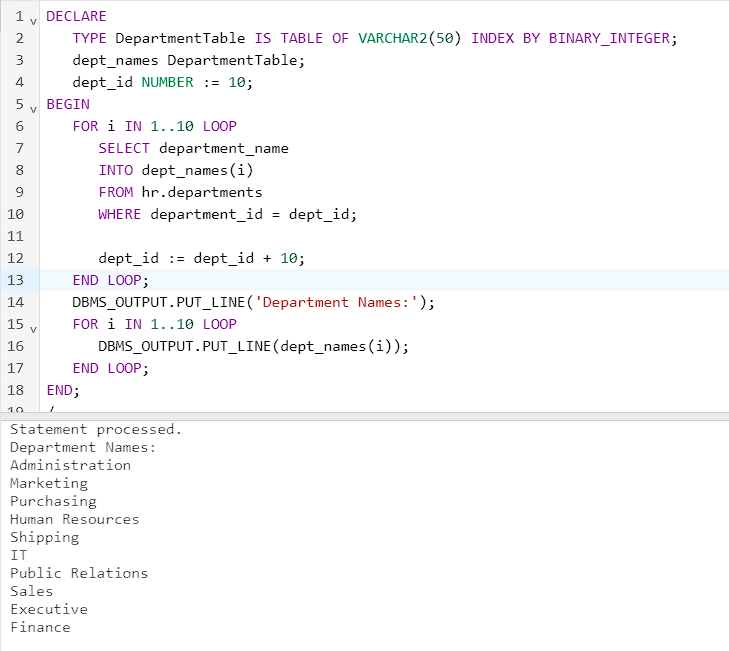
END;

/

4. 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 INDEX BY table. Using a loop, retrieve the names of 10 departments and store the names in the INDEX BY table. Start with department\_id 10. Increase 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 INDEX BY table. Using another loop, retrieve the department names from the INDEX BY table and display them.







DECLARE

TYPE DepartmentTable IS TABLE OF VARCHAR2(50) INDEX BY BINARY\_INTEGER;

dept\_names DepartmentTable;

dept\_id NUMBER := 10;

BEGIN

FOR i IN 1..10 LOOP

SELECT department\_name

INTO dept\_names(i)

FROM hr.departments

WHERE department\_id = dept\_id;

dept\_id := dept\_id + 10;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Department Names:');

FOR i IN 1..10 LOOP

DBMS\_OUTPUT.PUT\_LINE(dept\_names(i));

END LOOP;

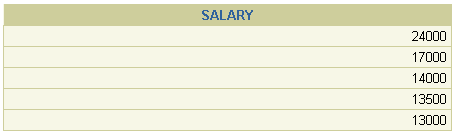
END;

/

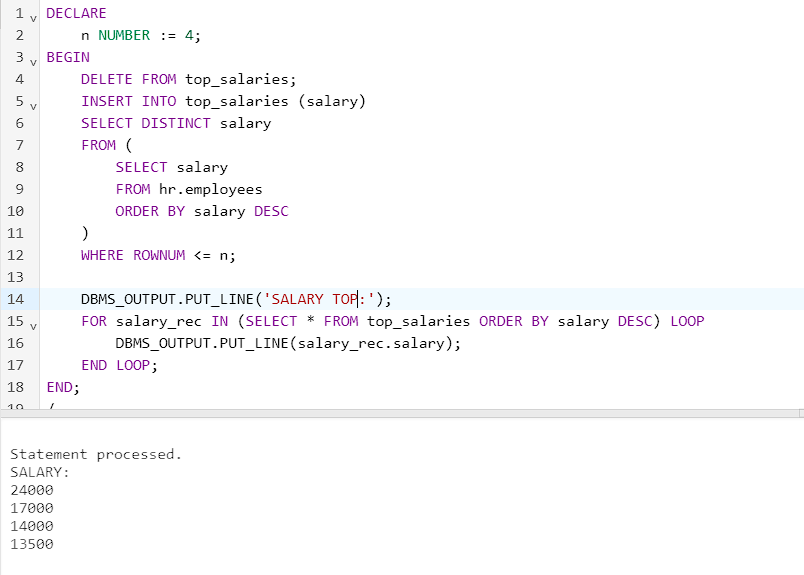
**Practice 4.2 (Explicit Cursors)**

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

* + - 1. Create a table top\_salaries with one column salary of the type NUMBER(8, 2).
      2. Declare a number *n* that represents the number of top *n* earners from the employees table.
      3. Declare a variable sal of type employees.salary.
      4. Declare a cursor, emp\_cursor, that retrieves the salaries of employees in descending order. Remember that the salaries should not be duplicated.
      5. 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.
      6. 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.



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



DECLARE

n NUMBER := 4;

BEGIN

DELETE FROM top\_salaries;

INSERT INTO top\_salaries (salary)

SELECT DISTINCT salary

FROM (

SELECT salary

FROM hr.employees

ORDER BY salary DESC

)

WHERE ROWNUM <= n;

DBMS\_OUTPUT.PUT\_LINE('SALARY TOP:');

FOR salary\_rec IN (SELECT \* FROM top\_salaries ORDER BY salary DESC) LOOP

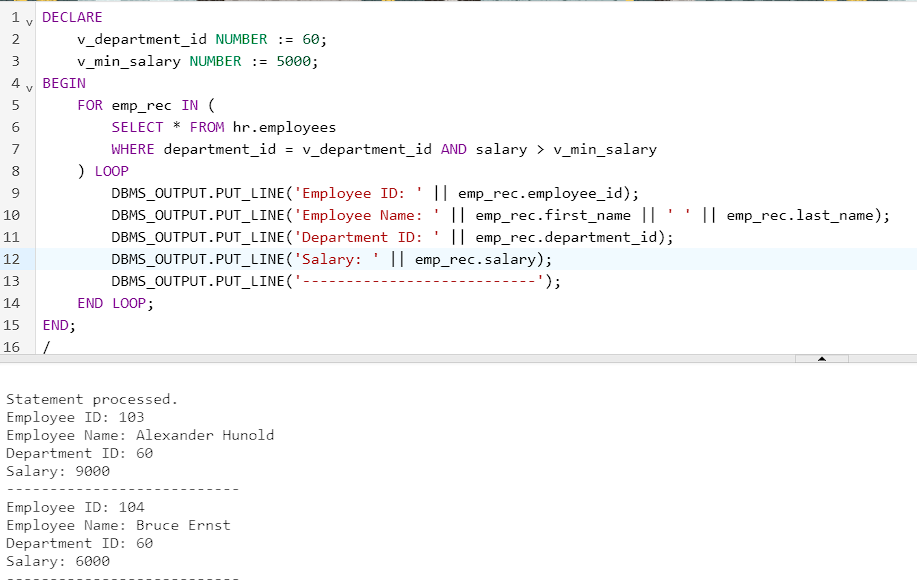
DBMS\_OUTPUT.PUT\_LINE(salary\_rec.salary);

END LOOP;

END;

/

1. Create a PL/SQL block that outputs all employees from the department 60 and with the salary more than 5000.



DECLARE

v\_department\_id NUMBER := 60;

v\_min\_salary NUMBER := 5000;

BEGIN

FOR emp\_rec IN (

SELECT \* FROM hr.employees

WHERE department\_id = v\_department\_id AND salary > v\_min\_salary

) LOOP

DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp\_rec.employee\_id);

DBMS\_OUTPUT.PUT\_LINE('Employee Name: ' || emp\_rec.first\_name || ' ' || emp\_rec.last\_name);

DBMS\_OUTPUT.PUT\_LINE('Department ID: ' || emp\_rec.department\_id);

DBMS\_OUTPUT.PUT\_LINE('Salary: ' || emp\_rec.salary);

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

END LOOP;

END;

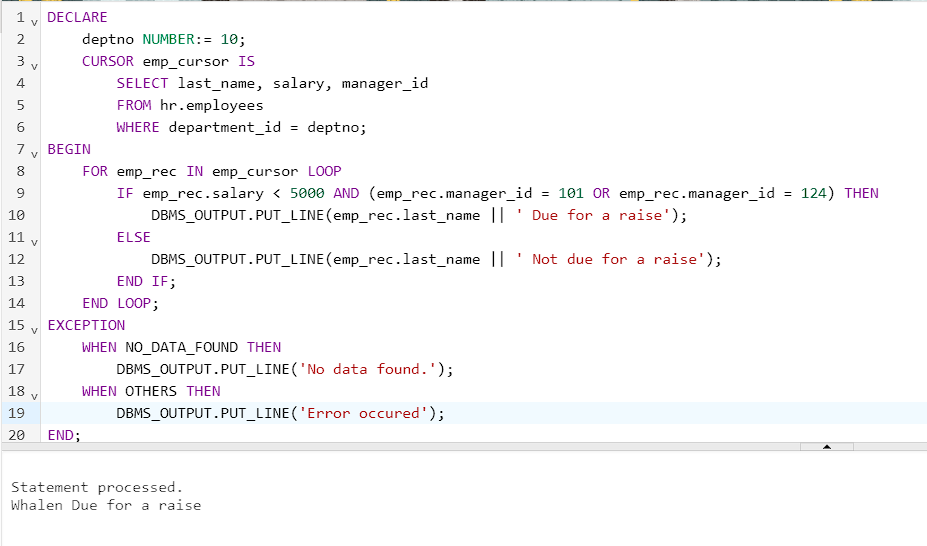
/

3. Rewrite the script of the task 2 using Cursor FOR Loop.

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

* + - 1. Define a variable deptno to provide the department ID.
      2. Declare a cursor, emp\_cursor, that retrieves the last\_name, salary, and manager\_id of the employees working in the department specified in 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 5000 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:



w

5. Write a PL/SQL block, which 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.

* + - 1. In the declarative section, declare a cursor dept\_cursor to retrieve department\_id, department\_name for those departments with department\_id less than 100. Order by department\_id.
      2. 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 with employee\_id of less than 120 and who work in that department.
      3. Declare variables to hold the values retrieved from each cursor. Use the %TYPE attribute while declaring variables.
      4. Open the dept\_cursor, use a simple loop and fetch values into the variables declared. Display the department number and department name.
      5. For each department, open the 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.
      6. Close all the loops and cursors, and end the executable section. Execute the script.

The sample output is shown below.

