Solution Lab N° 4

Exercise 1 :

```
SET SERVEROUTPUT ON;

BEGIN

  FOR i IN 1..10 LOOP

    IF MOD(i,2) = 0 THEN

      DBMS_OUTPUT.PUT_LINE(i || ' is even');

    ELSE

      DBMS_OUTPUT.PUT_LINE(i || ' is odd');

    END IF;

  END LOOP;

END;
```

Exercise 2 :

```
DECLARE

  v_salary NUMBER := 3000;

BEGIN

  IF v_salary < 2000 THEN

    DBMS_OUTPUT.PUT_LINE('Low salary');

  ELSIF v_salary BETWEEN 2000 AND 5000 THEN

    DBMS_OUTPUT.PUT_LINE('Average salary');

  ELSE

    DBMS_OUTPUT.PUT_LINE('High salary');

  END IF;

END;
```

Exercise 3:

```
DECLARE
  v_job_id VARCHAR2(10) := 'IT_PROG';
BEGIN
  CASE v_job_id
    WHEN 'IT_PROG' THEN
      DBMS_OUTPUT.PUT_LINE('Developer');
    WHEN 'ST_MAN' THEN
      DBMS_OUTPUT.PUT_LINE('Manager');
    WHEN 'SA_REP' THEN
      DBMS_OUTPUT.PUT_LINE('Sales');
    ELSE
      DBMS_OUTPUT.PUT_LINE('Other');
  END CASE;
END;
```

Exercise 4:

- The SELECT INTO statement automatically creates and manages a cursor behind the scenes.
- This is called an implicit cursor. Oracle handles the open, fetch, and close steps automatically.

Exercise 5:

```
DECLARE
  CURSOR c_emp IS
    SELECT first_name, last_name, salary
    FROM employees
    WHERE salary > 10000;

  v_first employees.first_name%TYPE;
  v_last  employees.last_name%TYPE;
  v_salary employees.salary%TYPE;
BEGIN
  OPEN c_emp;

  LOOP
    FETCH c_emp INTO v_first, v_last, v_salary;
    EXIT WHEN c_emp%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_first || ' ' || v_last ||' earns ' || v_salary || ' per month');
  END LOOP;

  CLOSE c_emp;
END;
```

Exercise 6:

```
CREATE OR REPLACE PROCEDURE show_employee_info(p_emp_id IN
employees.employee_id%TYPE) IS
  v_first employees.first_name%TYPE;
  v_job   employees.job_id%TYPE;
  v_salary employees.salary%TYPE;
BEGIN
  SELECT first_name, job_id, salary
  INTO v_first, v_job, v_salary
  FROM employees
  WHERE employee_id = p_emp_id;

  DBMS_OUTPUT.PUT_LINE('Name: ' || v_first);
  DBMS_OUTPUT.PUT_LINE('Job: ' || v_job);
  DBMS_OUTPUT.PUT_LINE('Salary: ' || v_salary);

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee not found');
END show_employee_info;
```

Exercise 7:

```
CREATE OR REPLACE FUNCTION get_annual_salary(p_emp_id IN
employees.employee_id%TYPE)
RETURN NUMBER IS
  v_salary employees.salary%TYPE;
  v_comm   employees.commission_pct%TYPE;
  v_annual NUMBER;
BEGIN
  SELECT salary, commission_pct
  INTO v_salary, v_comm
  FROM employees
  WHERE employee_id = p_emp_id;

  v_annual := (v_salary + NVL(v_comm, 0) * v_salary) * 12;
  RETURN v_annual;

EXCEPTION
  WHEN NO_DATA_FOUND THEN
    RETURN NULL;
END get_annual_salary;
```

Exercise 8:

```
DECLARE
  CURSOR c_emp IS
    SELECT first_name, last_name, salary
    FROM employees
    WHERE department_id = 60;

  v_first  employees.first_name%TYPE;
  v_last   employees.last_name%TYPE;
  v_salary employees.salary%TYPE;
BEGIN
  OPEN c_emp;

  LOOP
    FETCH c_emp INTO v_first, v_last, v_salary;
    EXIT WHEN c_emp%NOTFOUND;

    IF v_salary > 10000 THEN
      DBMS_OUTPUT.PUT_LINE(v_first || ' ' || v_last || ': High salary');
    ELSE
      DBMS_OUTPUT.PUT_LINE(v_first || ' ' || v_last || ': Normal salary');
    END IF;
  END LOOP;

  CLOSE c_emp;
END;
```

Exercises 9 and 10 introduce an advanced cursor concept, the **cursor FOR loop**, which can be implemented in two distinct forms, implicit and explicit.

1. **Implicit Cursor FOR Loop Syntax (introduced in exercise 9)**

```
BEGIN
  FOR record_variable IN (SELECT column1, column2, ...
                  FROM table_name
                  WHERE condition)
  LOOP
    -- use record_variable.column1, record_variable.column2, ...
  END LOOP;
END;
```

2. **Explicit Cursor FOR Loop Syntax (introduced in exercise 10)**

```
DECLARE
  CURSOR cursor_name IS
    SELECT column1, column2, ...
    FROM table_name
    WHERE condition;

BEGIN
  FOR record_variable IN cursor_name LOOP
    -- use record_variable.column1, record_variable.column2, ...
  END LOOP;
END;
```

Exercise 9:

```
BEGIN
  FOR rec IN (SELECT first_name, job_id FROM employees) LOOP
    DBMS_OUTPUT.PUT_LINE(
      rec.first_name || ': ' ||
      CASE rec.job_id
        WHEN 'SA_REP' THEN 'Sales Representative'
        WHEN 'IT_PROG' THEN 'Programmer'
        WHEN 'ST_MAN' THEN 'Store Manager'
        ELSE 'Other position'
      END);
  END LOOP;
END;
```

Here CASE is used as an expression not a statement, so we end it with END not END CASE;

Exercise 10:

```
CREATE OR REPLACE PROCEDURE increase_salary(p_dept_id IN
employees.department_id%TYPE) AS
  CURSOR c_emp IS
    SELECT employee_id, salary
    FROM employees
    WHERE department_id = p_dept_id;

  v_count NUMBER := 0;
BEGIN
  FOR rec IN c_emp LOOP
    UPDATE employees
    SET salary = salary * 1.10
    WHERE employee_id = rec.employee_id;

    v_count := v_count + 1;
  END LOOP;

  DBMS_OUTPUT.PUT_LINE('Total employees updated: ' || v_count);
END increase_salary;
```