EXERCISE 16

PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

```
DECLARE
  v_employee_id NUMBER := 110;
  v_salary employees.salary%TYPE;
  v_incentive NUMBER;
BEGIN
  SELECT salary INTO v_salary
  FROM employees
  WHERE employee_id = v_employee_id;

  -- Assuming incentive is 10% of the salary
  v_incentive := v_salary * 0.10;

  DBMS_OUTPUT.PUT_LINE('Incentive for employee ' || v_employee_id || ' is: ' ||
v_incentive);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
END;
```

PROGRAM 2

Write a PL/SQL block to show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

```
DECLARE
  "CaseSensitiveVar" VARCHAR2(30) := 'Hello';
  caseSensitiveVar VARCHAR2(30);
BEGIN
  -- This will raise an error because PL/SQL is case-insensitive and won't differentiate between the identifiers.
  caseSensitiveVar := "CaseSensitiveVar";
  DBMS_OUTPUT.PUT_LINE(caseSensitiveVar);
END;
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.

Sample table: employees

```
DECLARE
  v_employee_id NUMBER := 122;
  v_new_salary NUMBER := 5000;  -- Example of new salary
BEGIN
  UPDATE employees
  SET salary = v_new_salary
  WHERE employee_id = v_employee_id;

  IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Salary updated for employee ' || v_employee_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;
```

PROGRAM 4

Write a PL/SQL block to create a procedure using the "IS [NOT] NULL Operator"

and show AND operator returns TRUE if and only if both operands are TRUE.

```
CREATE OR REPLACE PROCEDURE check_null_and_operator (
  p_input1 IN VARCHAR2,
  p_input2 IN VARCHAR2
) IS
BEGIN
  IF p_input1 IS NOT NULL AND p_input2 IS NOT NULL THEN
    DBMS_OUTPUT.PUT_LINE('Both inputs are not null.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('One or both inputs are null.');
  END IF;
END;
/

BEGIN
  check_null_and_operator('Hello', 'World');  -- Both are not null
  check_null_and_operator(NULL, 'World');     -- One is null
END;
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
DECLARE
  v_name VARCHAR2(50);
BEGIN
  FOR rec IN (SELECT first_name FROM employees WHERE
first_name LIKE 'S%' ESCAPE '\') LOOP
    DBMS_OUTPUT.PUT_LINE('Name: ' || rec.first_name);
  END LOOP;
END;
```

PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num_small variable and large number will store in num_large variable.

```
DECLARE
  num1 NUMBER := 10;
  num2 NUMBER := 20;
  num_small NUMBER;
  num_large NUMBER;
BEGIN
  IF num1 < num2 THEN
    num_small := num1;
    num_large := num2;
  ELSE
    num_small := num2;
    num_large := num1;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Small number: ' || num_small || ', Large number: ' ||
num_large);
END;
```

PROGRAM 7

Write a PL/SQL procedure to calculate the incentive on a target achieved and

display the message either the record updated or not.

```
CREATE OR REPLACE PROCEDURE calculate_incentive (
  p_employee_id IN NUMBER,
  p_target_achieved IN NUMBER
) IS
  v_incentive NUMBER;
BEGIN
  v_incentive := p_target_achieved * 0.05;  -- Assuming 5% incentive
  UPDATE employees
  SET incentive = v_incentive
  WHERE employee_id = p_employee_id;
  IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Incentive updated for employee ' || p_employee_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;/
BEGIN
  calculate_incentive(110, 10000);  -- Example call
END;
```

PROGRAM 8

Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
CREATE OR REPLACE PROCEDURE calculate_incentive_by_sales (
  p_employee_id IN NUMBER,
  p_sales IN NUMBER
) IS
  v_incentive NUMBER;
BEGIN
  IF p_sales > 10000 THEN
    v_incentive := p_sales * 0.10;  -- 10% for sales over 10000
  ELSE
    v_incentive := p_sales * 0.05;  -- 5% for sales 10000 or less
  END IF;
  UPDATE employees
  SET incentive = v_incentive
  WHERE employee_id = p_employee_id;
  IF SQL%ROWCOUNT > 0 THEN
    DBMS_OUTPUT.PUT_LINE('Incentive updated for employee ' || p_employee_id);
  ELSE
    DBMS_OUTPUT.PUT_LINE('Employee not found.');
  END IF;
END;/

BEGIN
  calculate_incentive_by_sales(110, 15000);  -- Example call
END;
```

PROGRAM 9

Write a PL/SQL program to count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

```
DECLARE
  v_dept_id NUMBER := 50;
  v_emp_count NUMBER;
  v_vacancies NUMBER := 45;
BEGIN
  SELECT COUNT(*) INTO v_emp_count
  FROM employees
  WHERE department_id = v_dept_id;

  IF v_emp_count < v_vacancies THEN
    DBMS_OUTPUT.PUT_LINE('Department 50 has ' || (v_vacancies - v_emp_count) || ' vacancies.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Department 50 has no vacancies.');
  END IF;
END;
```

PROGRAM 10

Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

```
DECLARE
  v_dept_id NUMBER := 50;  -- Example department ID
  v_emp_count NUMBER;
  v_vacancies NUMBER := 45;
BEGIN
  SELECT COUNT(*) INTO v_emp_count
  FROM employees
  WHERE department_id = v_dept_id;

  IF v_emp_count < v_vacancies THEN
    DBMS_OUTPUT.PUT_LINE('Department ' || v_dept_id || ' has ' || (v_vacancies - v_emp_count) || ' vacancies.');
  ELSE
    DBMS_OUTPUT.PUT_LINE('Department ' || v_dept_id || ' has no vacancies.');
  END IF;
END;
```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates,

and salaries of all employees.

```
DECLARE
  CURSOR emp_cur IS
    SELECT employee_id, first_name || ' ' || last_name AS name, job_id, hire_date, salary
    FROM employees;
BEGIN
  FOR emp_rec IN emp_cur LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || emp_rec.employee_id || ', Name: ' || emp_rec.name ||
                         ', Job Title: ' || emp_rec.job_id || ', Hire Date: ' || emp_rec.hire_date ||
                         ', Salary: ' || emp_rec.salary);
  END LOOP;
END;
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department

names of all employees.

```
DECLARE
  CURSOR emp_cur IS
    SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, d.department_name
    FROM employees e
    JOIN departments d ON e.department_id = d.department_id;
BEGIN
  FOR emp_rec IN emp_cur LOOP
    DBMS_OUTPUT.PUT_LINE('ID: ' || emp_rec.employee_id || ', Name: ' || emp_rec.name ||
                         ', Department: ' || emp_rec.department_name);
  END LOOP;
END;
```

PROGRAM 13
Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all Jobs.

```
DECLARE
  CURSOR job_cur IS
    SELECT job_id, job_title, min_salary
    FROM jobs;
BEGIN
  FOR job_rec IN job_cur LOOP
    DBMS_OUTPUT.PUT_LINE('Job ID: ' || job_rec.job_id || ', Title: ' || job_rec.job_title ||
                         ', Min Salary: ' || job_rec.min_salary);
  END LOOP;
END;
```

PROGRAM 14
Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
DECLARE
  CURSOR emp_job_hist_cur IS
    SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, jh.start_date
    FROM employees e
    JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
  FOR rec IN emp_job_hist_cur LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id || ', Name: ' || rec.name ||
                         ', Job History Start Date: ' || rec.start_date);
  END LOOP;
END;
```

PROGRAM 15
Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

```
DECLARE
  CURSOR emp_job_hist_cur IS
    SELECT e.employee_id, e.first_name || ' ' || e.last_name AS name, jh.end_date
    FROM employees e
    JOIN job_history jh ON e.employee_id = jh.employee_id;
BEGIN
  FOR rec IN emp_job_hist_cur LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || rec.employee_id || ', Name: ' || rec.name ||
                         ', Job History End Date: ' || rec.end_date);
  END LOOP;
END;
```