

EXNO:11

PL SQL PROGRAMS

DATE:02.10.2024

PROGRAM 1

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

DECLARE

incentive NUMBER;

BEGIN

SELECT salary * 0.1 INTO incentive

FROM employees

WHERE employee_id = 110;

DBMS_OUTPUT.PUT_LINE('Incentive for Employee 110: ' || incentive);

END;

```
Incentive for Employee 110: 500
```

```
Statement processed.
```

PROGRAM 2

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

DECLARE

"MyVariable" NUMBER := 10; -- Quoted identifier (case-sensitive)

myvariable NUMBER := 20; -- Unquoted identifier (case-insensitive)

BEGIN

DBMS_OUTPUT.PUT_LINE('Value of "MyVariable": ' || "MyVariable");

DBMS_OUTPUT.PUT_LINE('Value of myvariable: ' || myvariable);

-- Attempting invalid case-insensitive reference

DBMS_OUTPUT.PUT_LINE('Incorrect reference to "MyVariable": ' || myVariable); -- This will cause an error

EXCEPTION

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);

END;

```
Value of "MyVariable": 10  
Value of myvariable: 20  
Incorrect reference to "MyVariable": 20
```

```
Statement processed.
```

```
0.09 seconds
```

PROGRAM 3

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

Sample table: employees

```
BEGIN
```

```
    UPDATE employees  
    SET salary = salary + 500  
    WHERE employee_id = 122;
```

```
COMMIT;
```

```
    DBMS_OUTPUT.PUT_LINE('Salary updated for employee ID 122');
```

```
EXCEPTION
```

```
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM);
```

```
END;
```

```
Salary updated for employee ID 122
```

```
1 row(s) updated.
```

```
0.01 seconds
```

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.

```
DECLARE
```

```
    PROCEDURE check_values(v1 IN VARCHAR2, v2 IN VARCHAR2) IS
```

```
    BEGIN
```

```
        IF v1 IS NOT NULL AND v2 IS NOT NULL THEN
```

```

        DBMS_OUTPUT.PUT_LINE('Both values are NOT NULL. AND condition is TRUE.');
```

ELSE

```

        DBMS_OUTPUT.PUT_LINE('AND condition is FALSE.');
```

END IF;

END;

BEGIN

```

    -- Example call to the procedure check_values('Hello',
    'World');    -- Both values are not NULL
    check_values('Hello', NULL);    -- One value is NULL
```

END;

```

Both values are NOT NULL. AND condition is TRUE.
AND condition is FALSE.
```

```

Statement processed.
```

```

0.01 seconds
```

PROGRAM 5

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

DECLARE

```

    v_text VARCHAR2(20) := '20% off';
```

BEGIN

```

    IF v_text LIKE '20\%%' ESCAPE '\' THEN
        DBMS_OUTPUT.PUT_LINE('Matches "20%" at the start');
    ELSIF v_text LIKE '_0%' THEN
        DBMS_OUTPUT.PUT_LINE('Second character is "0"');
    END IF;
```

END;

```

Matches "20%" at the start
```

```

Statement processed.
```

```

0.01 seconds
```

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; --
        Example value num2 NUMBER := 20; --
        Example value 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);
    DBMS_OUTPUT.PUT_LINE('Large number: ' || num_large);
END;
```

```
Small number: 10
Large number: 20

Statement processed.

0.00 seconds
```

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.

```
DECLARE
    PROCEDURE calculate_incentive(target IN NUMBER, actual_sales IN NUMBER) IS
        incentive NUMBER;
    BEGIN
        IF actual_sales >= target THEN incentive :=
            actual_sales * 0.1; -- 10% incentive
            DBMS_OUTPUT.PUT_LINE('Record updated with incentive: ' || incentive);
        ELSE
```

```

        DBMS_OUTPUT.PUT_LINE('Record not updated. Target not achieved.');
```

END IF;

```

END;
BEGIN
    -- Example call to the procedure
    calculate_incentive(1000, 1200); -- Target achieved
    calculate_incentive(1000, 800);  -- Target not achieved
END;
```

Record updated with incentive: 120
Record not updated. Target not achieved.

Statement processed.

PROGRAM 8

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

```

DECLARE
    PROCEDURE calculate_incentive(sales IN NUMBER) IS
        incentive NUMBER;
    BEGIN
        IF sales >= 1000 THEN incentive := sales * 0.1; -- 10%
            incentive for sales >= 1000
        ELSIF sales >= 500 THEN incentive := sales * 0.05; -- 5%
            incentive for sales >= 500
        ELSE
            incentive := 0; -- No incentive for sales < 500
        END IF;

        DBMS_OUTPUT.PUT_LINE('Incentive: ' || incentive);
    END;
BEGIN
    -- Example calls calculate_incentive(1200); -- High sales,
    10% incentive calculate_incentive(600); -- Medium
    sales, 5% incentive calculate_incentive(400); -- Low
    sales, no incentive
END;
```

```
Incentive: 120  
Incentive: 30  
Incentive: 0
```

```
Statement processed.
```

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 emp_count  
        NUMBER; vacancies  
        NUMBER := 45;  
BEGIN  
    -- Count the number of employees in department 50  
    SELECT COUNT(*) INTO emp_count  
    FROM employees  
    WHERE department_id = 50;  
  
    -- Check if there are vacancies  
    IF emp_count < vacancies THEN  
        DBMS_OUTPUT.PUT_LINE('There are vacancies in department 50.');    ELSE  
        DBMS_OUTPUT.PUT_LINE('No vacancies in department 50.');    END IF;  
END;
```

```
There are vacancies in department 50.
```

```
Statement processed.
```

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 dept_id NUMBER := 50; -- Example  
        department ID emp_count NUMBER;
```

```

total_vacancies NUMBER := 45; -- Total vacancies
in the department vacancies NUMBER;
BEGIN
    -- Count the number of employees in the specific department
    SELECT COUNT(*) INTO emp_count
    FROM employees
    WHERE department_id = dept_id;

    -- Calculate vacancies based on total vacancies and current employees
    vacancies := total_vacancies - emp_count;

    -- Check if there are vacancies
    IF vacancies > 0 THEN
        DBMS_OUTPUT.PUT_LINE('There are ' || vacancies || ' vacancies in department ' ||
dept_id);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No vacancies in department ' || dept_id);
    END IF;
END;

```

```

There are 43 vacancies in department 50
Statement processed.

```

PROGRAM 11

Write a PL/SQL program to display the employee IDs, names, job titles, hire dates, and salaries of all employees.

```

BEGIN
    FOR emp IN (SELECT employee_id, first_name, job_title, hire_date, salary
                FROM employees)
    LOOP
        DBMS_OUTPUT.PUT_LINE(emp.employee_id || ' ' || emp.first_name || ' ' || emp.job_title || '
' || emp.hire_date || ' ' || emp.salary);
    END LOOP;
END;

```

```
110 John Sales Rep 06/15/2015 5000
140 Mary Admin 07/20/2019 4000
122 Jane IT Specialist 08/25/2016 6000
130 Jim HR Manager 03/10/2018 6000
150 Emily Finance Clerk 01/30/2020 4500
```

PROGRAM 12

Write a PL/SQL program to display the employee IDs, names, and department names of all Employees.

```
BEGIN
  FOR emp IN (SELECT e.employee_id, e.first_name, d.department_name
               FROM employees e
               JOIN departments d ON e.department_id = d.department_id)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id ||
                          ', Name: ' || emp.first_name ||
                          ', Department: ' || emp.department_name);
  END LOOP;
END;
```

```
Employee ID: 130, Name: Jim, Department: HR
```

```
Statement processed.
```

```
0.01 seconds
```

PROGRAM 13

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

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



```
Job ID: IT_PROG, Title: IT Programmer, Min Salary: 4000
Job ID: MK_MAN, Title: Marketing Manager, Min Salary: 5000
Job ID: SA_REP, Title: Sales Representative, Min Salary: 2500
Job ID: FI_ACCOUNT, Title: Financial Accountant, Min Salary: 3500
Job ID: HR_REP, Title: HR Representative, Min Salary: 3000
```

```
Statement processed.
```

PROGRAM 14

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

```
BEGIN
  FOR emp IN (SELECT e.employee_id, e.first_name, j.start_date
               FROM employees e
               JOIN job_history j ON e.employee_id = j.employee_id)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id ||
                          ', Name: ' || emp.first_name ||
                          ', Job History Start Date: ' || emp.start_date);
  END LOOP;
END;
```

```
Employee ID: 122, Name: Jane, Job History Start Date: 08/25/2016
Employee ID: 110, Name: John, Job History Start Date: 06/15/2015
```

```
Statement processed.
```

PROGRAM 15

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

```
BEGIN
  FOR emp IN (SELECT e.employee_id, e.first_name, j.end_date
               FROM employees e
               JOIN job_history j ON e.employee_id = j.employee_id)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp.employee_id ||
```

```
        ', Name: ' || emp.first_name ||  
        ', Job History End Date: ' || emp.end_date);  
    END LOOP;  
END;
```

```
Employee ID: 122, Name: Jane, Job History End Date:  
Employee ID: 110, Name: John, Job History End Date: 06/15/2018  
  
Statement processed.
```