Ex. No:11 Roll No: 231901002

Date:09/11/2024 Name: ADHIYAN PERIYAR RP

PL SQL PROGRAMS

# PROGRAM 1

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

DECLARE incentive NUMBER;

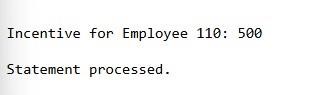
BEGIN

SELECT salary \* 0.1 INTO incentive

FROM employees

WHERE employee\_id = 110;

DBMS\_OUTPUT.PUT\_LINE('Incentive for Employee 110: ' || incentive); END;



# 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);

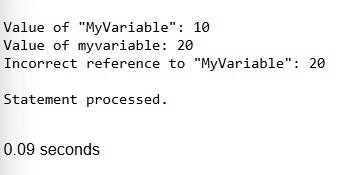
-- Attempting invalid case-insensitive reference

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;



# 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;

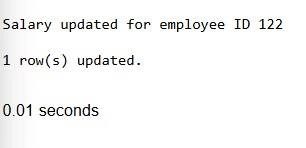
COMMIT;

DBMS\_OUTPUT.PUT\_LINE('Salary updated for employee ID 122');

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM); END;



# 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;

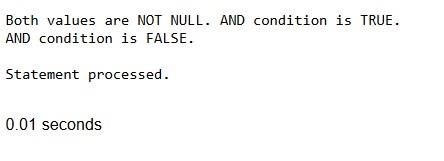
END;

BEGIN

-- Example call to the procedure check\_values('Hello',

'World'); -- Both values are not NULL check\_values('Hello', NULL); -- One value is NULL

END;



# 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

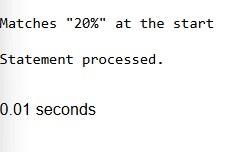
IF v\_text LIKE '20\%%' ESCAPE '\' THEN

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;



# 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

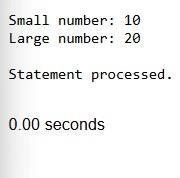
:= num1;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Small number: ' || num\_small);

DBMS\_OUTPUT.PUT\_LINE('Large number: ' || num\_large);

END;



# 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

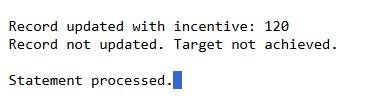
DBMS\_OUTPUT.PUT\_LINE('Record not updated. Target not achieved.'); END IF;

END;

BEGIN

-- Example call to the procedure calculate\_incentive(1000, 1200); -- Target achieved calculate\_incentive(1000, 800); -- Target not achieved

END;



# 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

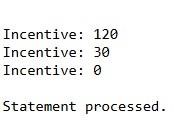
END IF;

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;



# 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;

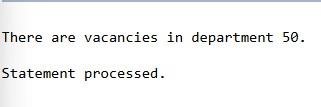
-- Check if there are vacancies

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;



# 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

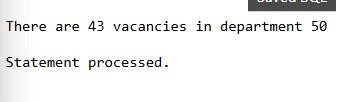
IF vacancies > 0 THEN

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;



# PROGRAM 11

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

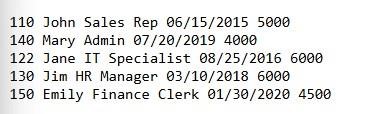
BEGIN

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;



# 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

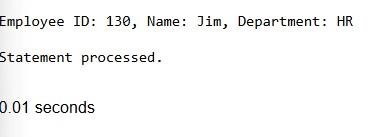
DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp.employee\_id ||

', Name: ' || emp.first\_name ||

', Department: ' || emp.department\_name);

END LOOP;

END;



# 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;



# 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

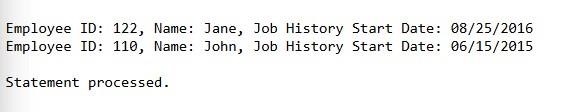
DBMS\_OUTPUT.PUT\_LINE('Employee ID: ' || emp.employee\_id ||

', Name: ' || emp.first\_name ||

', Job History Start Date: ' || emp.start\_date);

END LOOP;

END;



# 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;

