PL/SQL

Control Structures

In addition to SQL commands,PL/SQL can also process data usin flow of statements.the flow of control statements are classified into the following categories.

- Conditional control -Branching
- Iterative control - looping
- Sequential control

BRANCHING in PL/SQL:

Sequence of statements can be executed on satisfying certain condition .

If statements are being used and different forms of if are:

1.Simple IF

2.ELSIF

3.ELSE IF

SIMPLE IF:

Syntax:

```
IF condition THEN
   statement1;
   statement2;
END IF;
```

IF-THEN-ELSE STATEMENT:

Syntax:

```
IF condition THEN
   statement1;
ELSE
   statement2;
END IF;
```

ELSIF STATEMENTS:

Syntax:

```
IF condition1 THEN
   statement1;
```

ELSIF condition2 THEN

statement2;

ELSIF condition3 THEN

statement3;

ELSE

statementn;

END IF;

NESTED IF :

Syntax:

IF condition THEN

statement1;

ELSE

IF condition THEN

statement2;

ELSE

statement3;

END IF;

END IF;

ELSE

statement3;

END IF;

SELECTION IN PL/SQL(Sequential Controls)

SIMPLE CASE

Syntax:

CASE SELECTOR

WHEN Expr1 THEN statement1;

WHEN Expr2 THEN statement2;

ELSE

Statement n;

END CASE;

SEARCHED CASE:

CASE

WHEN searchcondition1 THEN statement1;

WHEN searchcondition2 THEN statement2;

:

:

ELSE

statementn;

END CASE;

ITERATIONS IN PL/SQL

Sequence of statements can be executed any number of times using loop construct.

It is broadly classified into:

- Simple Loop
- For Loop
- While Loop

SIMPLE LOOP

Syntax:

LOOP

statement1;

EXIT [WHEN Condition];

END LOOP;

WHILE LOOP

Syntax:

WHILE condition LOOP

statement1;

statement2;

END LOOP;

FOR LOOP

Syntax:

```
FOR counter IN [REVERSE]
   LowerBound..UpperBound
LOOP
statement1;
statement2;
END LOOP;
```

PROGRAM 1

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

```
DECLARE
    emp_id NUMBER := 110;
    salary NUMBER;
    incentive NUMBER;
BEGIN
    SELECT salary INTO salary FROM employees
    WHERE employee_id = emp-id;
    incentive := salary * 0.10;
    DBMS_OUTPUT.PUT_LINE ('Incentive: '|| 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" VARCHAR(20) := 'Hello';
BEGIN
      DBMS_OUTPUT.PUT_LINE("~~Greeting~~ myVariable");
      DBMS_OUTPUT.PUT_LINE(MyVariable);
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 + 2000
    WHERE employee_id = 122;
DBMS.output.put_line("salary updated for employee, 22);
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_VALUES()

a NUMBER := 10;
 b NUMBER := 20;

BEGIN
    if a is NOT NULL and b is not NULL then
    DBMS_OUT.PUT_LINE (Both are not NULL ,TRUE');
ELSE
    DBMS_OUTPUT.PUT_LINE ('FALSE');

END IF;

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(20) := 'abc_123';
BEGIN
        if V_name LIKE 'abc\_%' Escape'\' then
        DBMS_OUTPUT.Put_line ('Matched');
        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
        a   NUMBER := 25;
        b   NUMBER := 10;
        num_small  Number;
        num_large  number;
BEGIN
        if  a <b Then
        num_small := b;
        num_large := a;

END IF;
        DBMS_OUTPUT.PUT_LINE ('SMALL =' ||num_small);
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 call_incentive (p_id
number)
is
    v_rows Number;
BEGIN
    update employees
    set salary = salary + (salary * 0.10)
    WHERE employee_id = p_id;
If v_rows > 0 then
    Dbms_output.put_line ('Record updated');
else
    Ddms_output_Put_line ('No record');
END;
```

PROGRAM 8

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

```
CREATE or replace procedure calc-incentive
(p-sales NUMBER) is
    v_incentive number;
BEGIN
    if p-sales >100000 then
    v_incentive := p-sales * 0.10;
ELSE
    v_incentive := p-sales * 0.01;
END IF;
    DBMS_OUTPUT.PUT_LINE ('Incentive =
    calc-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
      SELECT  COUNT(*) INTO emp_count FROM employees
                    WHERE department_Pd  =50;
       IF   emp_count  < vacancies  THEN
             DBMS_ OUTPUT. PUT_LINE ('vacancies  available: '||
                          (vacancies - emp_count));
              ELSE
                    DBMS_OUTPUT. PUT_LINE ('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
        dept_id NUMBER := 60;
        emp_count NUMBER;
        max_vacancies NUMBER := 60;
BEGIN
        SELECT COUNT (*) INTO emp_count FROM employees
           WHERE department_id = dept_id;
        IF emp_count < max_vacancies THEN
                DBMS_OUTPUT. PUT_LINE ('Vacancies: ' || (max_vacancies
                                                        emp_count));
         ELSE
                DBMS_OUTPUT. PUT_LINE ('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.

```
BEGIN
    FOR rec IN (SELECT employee_id, first_name ||
        ' ' || last_name AS name, job_id, hire_date,
        salary FROM employees)
    LOOP
        DBMS_OUTPUT.PUT_LINE (rec.employee_id || ' '
        || rec.name || '-' || rec.job_id || '-' ||
        rec.hire_date || '-' || rec.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  rec IN C
         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
    )
    Loop
         DBMS_OUTPUT.PUT_LINE(rec.employee_id ||
         '_' || rec.name || '_' || 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.

```
BEGIN
    FOR rec IN (SELECT job_id, job_title, min_salary
      FROM jobs)
     LOOP
        DBMS_OUTPUT.PUT_LINE (rec.job_id || '-' ||
        rec.job_title || '-' || 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.

```
BEGIN
    FOR   rec IN(
        SELECT   e.employee_id , e.first_name || ' ' ||
        e.last_name AS name, jh.start_data
        FROM employees e
        JOIN job_history jh ON e.employee_id = jh.employee_
                                                    id
        )
    LOOP
        DBMS_OUTPUT.PUT_LINE(rec.employee_id || '_'
        || rec.name || '_' || rec.start_data);
    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 rec IN (
        SELECT e.employee_id, e.first_name || ' ' ||
        e.last_name AS name, jh.end_data FROM
        employeese.
        JOIN job_history jh ON e.employee_id ||
            jh.employee_id)
LOOP
    DBMS_OUTPOT.PUT_LINE (rec.employee_id || '-'
    || rec.name ||
    '-' || rec.end_dat
        );
END LOOP;
END;
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	5
Program/Execution (5)	5
Viva(5)	5
Total (15)	15
Faculty Signature	