PL/SQL

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
  v_Salary employees.salary %TYPE;
  v_incentive NUMBER;
BEGIN
  select salary INTO v_salary FROM employees
  where employee_id = 110;
  v_incentive := v_Salary * 0.10;
  DBMS_output.PUT_LINE ('Incentive: ' || v_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
   v_test VARCHAR2(20) := 'unquoted';
   "My-Variable" VARCHAR2(20) := 'Quoted';
BEGIN
   DBMS_OUTPUT.PUT_LINE ('Valid unquoted: ' || v_TEST);
   DBMS_OUTPUT.PUT_LINE ('Valid Quoted: ' || "My-Variable");
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 * 1.05
   where employee_id = 122;
   if SQL%ROWCOUNT > 0 Then
     commit;
     DBMS_output.Put_line (' Salary adjusted ');
   endif;
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
  v_name VARCHAR2(250) := 'John';
  v_commission Number := NULL;
BEGIN
    if v_commission is NULL Then
        DBM_output.put_line ('commission is NULL.');
    End if

    if (v_name is NOT NULL) AND (1=1) Then
        DBMS_Output.put_line ('AND operator returned True');
    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_string varchar(50) := 'Test file_20%';
BEGIN
      -- % and _ wildcards
      If vstring LIKE 'Test %_%' THEN
            DBMS_OUTPUT.PUT_Line ('Matches with % and _.');
End If;
If v_string Like '%20 %%' ESCAPE '\' Then
    Dbms_output.put_line ('Matches literal % using escape');
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
  num_a  Number := 55;
  num_b  Number := 12;
  num_small NUMBER;
  num_large Number;

BEGIN
  If num_a > num_b Then
       num_large := num_a;
       num_small := num_b;
  else
       num_large := num_b;
       num_small := num_a;
  end if;
  DBMS_output.Put_Line ('small: ' num_small ' large: '
  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.

```
BEGIN
  If target_achieved >= 10000 THEN
     Update employees
     Set incentive_pay = NVL (incentive_pay, 0) + 500
  where employee_id := & employee_id;

  if SQL %ROWCOUNT > 0 then
    Commit;
    dbms_output.put_line('Record not updated (ID not found)');
  End if;

  else
    DBMS_Output.put_Line ('Record No update. Target not met');
  End if;

End;
/
```

PROGRAM 8

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

```
DECLARE
    v_sale Number := &monthly_Sales;
    v_incentive Number := 0;
BEGIN
  if v_Sales = 30000 Then v_incentive := 2000;
  ELSIF v_Saless = 10000 Then v_ incentive := 500;
  END IF;

  IF v_incentive > 0 THEN
      UPDATE employee SET incentive_pay = NVL (incentive_pay,0)+
             v_incentive where employee_id = &employee_id;

      commit;

  endif;
  DBMS_Output.put_Line ('Incentive Awarded: ' ||v_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
    v_current_emp NUMBER;
    v_vacancies constant Number := 45;
BEGIN
    Select count(*) INTO v_current_emp FROM employees
    where department_id = 50;
    If v_vacancies>0 THEN
      DBMS_Output_LINE('Dept 50 has ' || v_currentemp
      || 'employees.');
      DBMS_output_Line ('Result': 'yes,' || v_vacancies
        || 'vacancies available');
    ELSE
      DBMS_output.put_Line ('Result: 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 := &dept_id;
    v_Capacity CONSTANT NUMBER := 60;
    v_current_emp NUMBER;
    v_Vacancies NUMBER;
BEGIN
    select count(*) into v_current_emp from employees
    where department_id = v_dept_id;
    v_Vacancies := v_Capacity - v_current_emp;
    DBMS_output.put_Line('Employees in Dept ' || v_dept_id || ':'
                         || v_current_emp);
    IF v_Vacancies > 0 THEN
        DBMS_output.put_Line('Result: Yes', || v_Vacancies ||
                             ' vacancies');
    ELSE
        DBMS_output.put_Line('Result : No vacancies or
                              over capacity.');
    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 r_emp IN (
        Select e.employee_id, e.first_name ||' ' ||
  e.last_name AS name, j.job_title, e.hire_date, e.salary

  FROM employees e JOIN jobs j ON e.job_id = j.job_id
      order by e.employee_id
  )
  LOOP
  DBMS_Output.put_Line (r_emp.employee_id || ' ' || r_emp.name
  || ' ' || r_emp.job_title || ' ' || r_emp.hire_date || ' ' ||
   r_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 r_emp IN (
     SELECT e.employee_id, e.first_name || ' ' || e.last_name
  AS name, d.department_name
     FROM employees e LEFT JOIN departments d
  ON e.department_id = d.department_id ORDER BY
                        e.employee_id
  )
  LOOP
     DBMS_output.PUT_LINE (r_emp.employee_id
  || ' ' || r_emp.name || ' ' || r_emp.name || ' ' ||
        r_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 r_job IN (
        Select job_id, job_title, main_salary
        From jobs
        order by min salary DESC
    )
    Loop
        DBMS_Output.put_Line (r_job.job_id || ' ' ||
        r_job.job_title || ' ' || r_job. im_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 r_hist IN (
        Select e.employee_id, e.first_name || ' ' || e.last_name
        as name, jh.start_date  from employees e join
    job_history jh ON employee_id = jh.employee_id
        order by eemployee_id , jh.start_date
    )
    Loop
      DBMS_output.put_Line ( r_hist.employee_id || ' ' ||
      r_hist.name || ' ' || r_hist.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 r_hist IN (
        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
      order by e.employee_id, jh.end_date
    )
    LOOP
      DBMS_OUTPUT.PUT_LINE (r_hist.employee_id || ' ' ||
      r_hist.name || ' END: ' || NVL (TO_CHAR( r_hist.end_date)
        , 'N/A'));
   END LOOP;
END;
/
```

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