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;
```

PROGRAM 1

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

```
Set serveroutput on;
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 ("Employee salary :' ||
                          v_salary);
DBMS_OUTPUT.PUT_LINE ("Incentive(10%.) :' ||
                          TO_CHAR(v_incentive));
exception
   when no_data_found then
        DBMS_OUTPUT.PUT_LINE ('No emp. with id 110');
   when others then
        DBMS_OUTPUT.PUT_LINE ('Error: ' ||SQLERRM);
END;
```

PROGRAM 2

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

```
Begin
    Execute immidiate 'BEGIN EXECUTE IMMIDIATE'
          'DROP TABLE qi_test '' ;
    Exception when others then NULL;
    END; ';
    Execute immidiate ' CREATE TABLE qi_test
     ("MyCol" Varchar2(20))';
    Execute immidiate 'Insert into qi_test ("MyCol"
     values (''x'');
    Begin
     - Execute immidiate 'select mycol from
          qi_test' INTO:dummy ;
     Exception when others then
     DBMS_OUTPUT.PUT_LINE('ExpectedError :' ||
        SQLERRM);
     END;
     Execute immidiate 'Drop table qi_test';
     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 + 5000
     where employee_id = 122;
    DBMS_OUTPUT.PUT_LINE (SQL%.ROWCOUNT ||
        'Rows updated');
     COMMIT;
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 p-check (a IN
  varchar2, b IN varchar2) IS
Begin
   If a is NOT NULL and b IS NOT NULL then
   DBMS_OUTPUT.PUT_LINE ('Not both 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
    V1 Number;
Begin
    select count(*) into V1 from employees
    where first_name like 'S%';
    DBMS_OUTPUT.PUT_LINE('Starts S:' || V1);
    Execute immidiate 'CREATE TABLE t_like (s varchar2(5
    Execute immidiate q'INSERT into t_like
    values ('50% off')';
    select count(*) into V1 from t_like where like
    '%\%%' escape '\';
    DBMS_OUTPUT.PUT_LINE('Contains %:' || V1);
    Execute immediate 'DROP Table t_like';
    Exception when others then NULL;
    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 := &num1; num2 Number :=
  &num2;
 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 = ' || num_small ||
 'Large = ' || num_large);
END;
```

PROGRAM 8

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

```
Create or replace procedure slab_incent
(p_emp in number, p_sales in number) IS
pct number := case when p_sales < 10000
then 2 when p_sales < 25000 then 5 when
P_sales < 50000 then 8 else 12 end;
 sal NUMBER; inc number;
Begin
   Select salary into sal from employees
   where employee_id = p_emp;
   inc := sal * pct/100;
   DBMS_OUTPUT.PUT_LINE('Incentive' || inc ||
    ' ('||pct||'%.)');
exception when no_data_found then
DBMS_OUTPUT.PUT_LINE('No emp');
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
    cnt number;
    vacancies constant number := 45;
Begin
    select count(*) into cnt from employees
    where department_id = 50;
    DBMS_OUTPUT.PUT_LINE('Count = '||cnt||',
    Vacancies = '||(vacancies - cnt));
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.

```
Create or replace procedure dept_vac
 (p_dept in number, p_total in number) IS
    cnt number ;
Begin
    Select count (*) into cnt from employees
    where department_id = p_dept ;
    If p_total-cnt >0 then
DBMS_OUTPUT.PUT_LINE ('Vacancies = ' || p_tot-
                                        cnt);
elsif p_total-cnt =0 then
DBMS_OUTPUT.PUT_LINE('No vacancies') ;
 else DBMS_OUTPUT.PUTLINE ('Over staffed')
 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 in (select e.employee_id, NVL(e.first_
                                        name,'')||
   ' '||NVL(e.last_name,'') name, j.job_title,
   e.hire_date, e.salary from employees e
   left join jobs j using (job_id))
   LOOP
DBMS_OUTPUT.PUT_LINE(r.employee_id||' '||
r.name||' '||NVL(r.job_title,'_')||' '||
TO_CHAR(r.hire_date,'YYYY-MM-DD')||' '||
 NVL(TO_CHAR(r.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 in (select e.employee_id, NVL(e.first_name,'')
    ||' '||NVL(e.last_name,'') name, d.department_
                                          name
    from employee e left join departments d
    using (department_id))
    LOOP
  DBMS_OUTPUT_PUT_LINE(r.employee_id ||' '||
    r.name||' '||NVL(r.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 in (select job_id, job_title, min_salary
  from jobs order by job_id ).LOOP
    DBMS_OUTPUT.PUT_LINE( r.job_id || ' ' ||
     r.job_title || ' ' || NVL (TO_CHAR (r.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 r in (select e.employee_id, NVL(e.first_name
    ,' ') || ' ' || NVL(e.last_name,' ') name,
    jh.start_date from job_history jh Join
    employee e using(employee_id) order by
    e.employee_id, jh.start_date)
    LOOP
DBMS_OUTPUT.PUT_LINE(r.employee_id || ' ' ||
r.name || ' ' ||
TO_CHAR(r.start_date,'YYYY-MM-DD'));
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 in (select e.employee_id, NVL(e.first_name,'')
    ||' '|| NVL(e.last_name,' ') name, jh.end_date from
    job_history jh JOIN employees e using (employee_id)
    order by e.employee_id, jh.end_date)
     LOOP
    DBMS_OUTPUT.PUT_LINE (r.employee_id ||' '|| ||
    r.name||' '|| TO_CHAR (r.end_date,'YYYY-MM-DD'));
    END LOOP;
     END;
```

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