Exercise Number: 7

Title of the Exercise : CONTROL STRUCTURE

Date of the Exercise :

OBJECTIVE (AIM) OF THE EXPERIMENT

To create PL/SQL programs to implement various types of control structure.

FACILITIES REQUIRED AND PROCEDURE

a) Facilities required to do the experiment:

Sl.No.	Facilities required	Quantity
1	System	1
2	Operating System	Windows
3	Front end	
4	Back end	Oracle11g

b) PL/SQL Syntax:

PL/SQL can also process data using flow of statements. The flow of control statements are classified into the following categories.

- Conditional control –Branching
- Iterative control – looping
- Sequential control - Selection

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. If then else 3. Else if 4. Nested if

SELECTION IN PL/SQL (Sequential Controls)

1. Simple case 2. Searched case

ITERATIONS IN PL/SQL

Sequence of statements can be executed any number of times using loop construct. It is broadly classified into:

1.Simple Loop 2. For Loop 3. While Loop

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
```

statement;
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:

Syntax:

CASE
WHEN searchcondition1 THEN statement1;
WHEN searchcondition2 THEN statement2;
::
ELSE
statementn;
END CASE;

ITERATIONS IN PL/SQL

SIMPLE LOOP

Syntax:

LOOP
statement1;
EXIT [WHEN Condition];
END LOOP;

Example:

```
Declare
A number:=10;
Begin
Loop
a := a+25;
exit when a=250;
end loop;
dbms_output.put_line(to_char(a));
end;
/
```

WHILE LOOP

Syntax

```
WHILE condition LOOP
statement1;
statement2;
END LOOP;
```

Example:

```
Declare
i number:=0;
j number:=0;
begin
while i<=100 Loop
j := j+i;
i := i+2;
end loop;
dbms_output.put_line('the value of j is' ||j);
end;
/
```

FOR LOOP

Syntax:

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

Example:

```
Begin
For I in 1..2
Loop
Update emp set field = value where condition;
End loop;
End;
/
```

Q1: write a pl/sql program to swap two numbers

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Declare three variables and read variables through a and b
2	Swap the values of a and b using temporary variables
3	Display the swapped results

d) Program:

```
SQL>edit swapping.sql
declare
a number(10);
b number(10);
c number(10);
begin
dbms_output.put_line('THE PREV VALUES OF A AND B WERE');
dbms_output.put_line(a);
dbms_output.put_line(b);
```

```
a:=&a;
b:=&b;
c:=a;
a:=b;
b:=c;
dbms_output.put_line('THE VALUES OF A AND B ARE');
dbms_output.put_line(a);
dbms_output.put_line(b);
end;
```

e)output:

```
SQL> @ swapping.sql
19 /
Enter value for a: 5
old 6: a:=&a;
new 6: a:=5;
Enter value for b: 3
old 7: b:=&b;
new 7: b:=3;
THE PREV VALUES OF A AND B WERE
53
THE VALUES OF A AND B ARE
35
```

PL/SQL procedure successfully completed.

Q2: Write a pl/sql program to find the largest of three numbers

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read three numbers through a, b & c
2	Find the biggest among three using nested if statement
3	Display the biggest no as result

d)Program:

```
SQL>set server output on;
SQL>edit biggest.sql
declare
a number;
b number;
c number;
begin
a:=&a;
b:=&b;
c:=&c;
if a>b then
if a>c then
dbms_output.put_line ('biggest is:' ||to_char(a));
else
dbms_output.put_line('biggest is :' ||to_char(c));
end if;
elsif b>c then
dbms_output.put_line('biggest is :' ||to_char(b));
else
dbms_output.put_line('biggest is :' ||to_char(c));
```

```
end if;
end;
```

e)output:

```
SQL>@biggest.sql
/
        Enter value for a: 5
        old 6: a:=&a;
        new 6: a:=5;
        Enter value for b: 5
        old 6: b:=&b;
        new 6: b:=8;
        Enter value for c: 8
        old 6: c:=&c;
        new 6: c:=4;
        biggest is : 8
```

Q3: write a pl/sql program to find the total and average of 6 subjects and display the grade

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read six numbers and calculate total and average
2	Find whether the student is pass or fail using if statement
3	Find the grade using nested elseif statement
4	Display the Grade, Percentage and Total of the student

d)Program:

```
SQL> edit grade.sql
        declare
        java number(10);
        dbms number(10);
        co number(10);
        se number(10);
        es number(10);
        ppl number(10);
        total number(10);
        avgs number(10);
        per number(10);
        begin
        dbms_output.put_line('ENTER THE MARKS');
        java:=&java;
        dbms:=&dbms;
        co:=&co;
        se:=&se;
        es:=&es;
        ppl:=&ppl;
        total:=(java+dbms+co+se+es+ppl);
        per:=(total/600)*100;
        if java<50 or dbms<50 or co<50 or se<50 or es<50 or ppl<50 then
        dbms_output.put_line('FAIL');
        if per>75 then
        dbms_output.put_line('GRADE A');
        elsif per>65 and per<75 then
        dbms_output.put_line('GRADE B');
        elsif per>55 and per<65 then
```

```
dbms_output.put_line('GRADE C');
else
dbms_output.put_line('INVALID INPUT');
end if;
dbms_output.put_line('PERCENTAGE IS '||per);
dbms_output.put_line('TOTAL IS '||total);
end;
```

e)output:

```
SQL> @ grade.sql
31 /
Enter value for java: 80
old 12: java:=&java;
new 12: java:=80;
Enter value for dbms: 70
old 13: dbms:=&dbms;
new 13: dbms:=70;
Enter value for co: 89
old 14: co:=&co;
new 14: co:=89;
Enter value for se: 72
old 15: se:=&se;
new 15: se:=72;
Enter value for es: 76
old 16: es:=&es;
new 16: es:=76;
Enter value for ppl: 71
old 17: ppl:=&ppl;
new 17: ppl:=71;
GRADE A
PERCENTAGE IS 76
TOTAL IS 458
```

PL/SQL procedure successfully completed.

Q4: Write a pl/sql program to find the sum of digits in a given number

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number. Separate the digits using modular function
2	Sum the digits separated by mod function
3	Display the sum of digits

d)Program:

```
SQL>edit sumofdigits.sql
declare
a number;
d number:=0;
sum1 number:=0;
begin
a:=&a;
while a>0
loop
d:=mod(a,10);
sum1:=sum1+d;
```

```
a:=trunc(a/10);
end loop;
dbms_output.put_line('sum is'|| sum1);
end;
```

e)output:

```
SQL> @ sumofdigits.sql
        16 /
```

Q5: write a pl/sql program to display the number in reverse order

c)Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number. Separate the digits using modular function
2	Reverse the digits separated by taking remainder from mod function
3	Display the reverse of the digits

d)Program:

```
SQL>edit reverse.sql
declare
a number;
rev number;
d number;
begin
a:=&a;
rev:=0;
while a>0
loop
d:=mod(a,10);
rev:=(rev*10)+d;
a:=trunc(a/10);
end loop;
dbms_output.put_line('no is'|| rev);
end;
```

e)output:

```
SQL> @ reverse.sql
16 /
Enter value for a: 536
        old 6: a:=&a;
        new 6: a:=536;
no is 635
PL/SQL procedure successfully completed.
```

Q6: Write a PL / SQL program to check whether the given number is prime or not

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read the number
2	Using mod function find the given number is prime or not
3	Display the result

d)Program:

```
SQL>edit prime.sql
declare
a number;        c number:=0;              i number;
begin
a:=&a;
for i in 1..a
loop
if mod(a,i)=0 then
c:=c+1;
end if;
end loop;
if c=2 then
dbms_output.put_line(a ||'is a prime number');
else
dbms_output.put_line(a ||'is not a prime number');
end if;
end;
```

e)output:

```
SQL> @ prime.sql
19 /
Enter value for a: 11
old 6: a:=&a;
new 6: a:=11;
11is a prime number
PL/SQL procedure successfully completed.
```

Q7: Write a PL/SQL program to find the factorial of a given number

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Read a number for calculating factorial value.
2	Calculate the factorial of a given number using for loop
3	Display the factorial value of a given number.

d)Program:

```
SQL>edit fact.sql
declare
n number;f number:=1;
begin
n:=&n;
for i in 1..n
loop
f:=f*i;
end loop;
dbms_output.put_line('the factorial is'|| f);
end;
```

e)output:

```
SQL> @ fact.sql
12 /
Enter value for n: 5
old 5: n:=&n;
new 5: a:=5;
the factorial is 120
```

Q8: write a pl/sql code block to calculate the area of a circle for a value of radius varying from 3 to 7. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns radius & area

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Create a table named areas with radius and area
2	Initialize values to pi, radius and area
3	Calculate the area using while loop. Display the result.

d)Program:

```
SQL> create table areas(radius number(10),area number(6,2));
Table created.
PROGRAM
declare
pi constant number(4,2):=3.14;
radius number(5):=3; area number(6,2);
begin
while radius<7
loop
area:=pi*power(radius,2);
insert into areas values(radius,area);
radius:=radius+1;
end loop;
end;
```

e)output:

```
SQL> @ AREAOFCIRCLE.SQL
13 /
PL/SQL procedure successfully completed.
SQL> SELECT * FROM AREAS;
RADIUS AREA
---------- ----------
3 28.26
4 50.24
5 78.5
6 113.04
```

Q9: write a PL/SQL code block that will accept an account number from the user, check if the users balance is less than minimum balance, only then deduct rs.100/- from the balance. This process is fired on the acct table.

c) Procedure for doing the experiment:

Step no.	Details of the step
1	Develop a query to Create the table acct and insert values into them
2	Develop a PL/SQL program to read the account number.
3	Check the balance for the account no. check if the users balance is less than minimum balance, only then deduct rs.100/- from the balance
4	Update the balance changes into the acct table.

d)Program:

```
SQL> create table acct(name varchar2(10),cur_bal number(10),acctno number(6,2));
SQL> insert into stud values('&sname',&rollno,&marks);
SQL> select * from acct;
ACCTNO NAME CUR_BAL
---------- ---------- ----------
777 sirius 10000
765 john 1000
855 sam 500
353 peter 800
declare
mano number(5);
mcb number(6,2);
minibal constant number(7,2):=1000.00;
fine number(6,2):=100.00;
begin
mano:=&mano;
select cur_bal into mcb from acct where acctno=mano;
if mcb<minibal then
update acct set cur_bal=cur_bal-fine where acctno=mano;
end if;
end;
```

e)output:

```
SQL> @ BANKACC.sql
13 /
Enter value for mano: 855
old 7: mano:=&mano;
new 7: mano:=855;
        PL/SQL procedure successfully completed.
```

f)Result:

Thus the above creation of PL/SQL programs to implement various types of control structure was successfully executed.

QUESTIONS AND ANSWERS

1. What is meant by 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. If then else 3. Else if 4. Nested if

2. What are selection statements?

1. Switch case statement

3. Define iterations IN PL/SQL

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

4. Classify the iteration statements `in PL/SQL

It is broadly classified into:

1.Simple Loop
2. For Loop
3. While Loop