**PL/SQL:**

1. It stands for procedural language/structure query language in SQL we can’t execute the same statement repeatedly to a particular no of times but where as in PL/SQL we can execute the same statement repeatedly because it supports looping mechanism.
2. In SQL we can’t execute more than one statement at a time. But where as in PL/SQL we can execute more than one statement concurrently.
3. PL/SQL program is a combination of procedural language statements and structure query language statements.
4. All PL statements are executed by PL engine where as all the SQL statements are executed by PL engine.
5. PL/SQL supports all the principles of procedure language such as procedural functions, control statements, conditional statements e.t.c., and also it supports some principles of OOPS.
6. PL/SQL is not a consecutive language and every statement of PL/SQL program should ends with ;(semicolon).
7. PL/SQL is the high performance transaction processing language.

**Data types in PL/SQL:**  The data types which are using in SQL same data types are supported in PL/SQL.

**Operators in PL/SQL:** The operators which are using in SQL same operators are by PL/SQL except assignment operator.

**Input statements in PL/SQL:**

1. There is no input statements in PL/SQL to input the values at run time.
2. If we want input values at run time then we use insertion operator (&).

**Output statements in PL/SQL:**

Dbms\_output.put\_line()/ dbms\_output.put()

This is other statement in PL/SQL which is used to print the values on the standard output device.

**SYN:** dbms\_output.put\_line(‘message’);

**EX:** dbms\_output.put\_line(‘Sathya Technologies’);

**SYN:** dbms\_output.put\_line(‘message’||variable);

**EX:** dbms\_output.put\_line(‘sum is ‘||C);

**SYN:** dbms\_output.put\_line(Variable);

**EX:** dbms\_output.put\_line©;

**Variable declaration in PL/SQL:**

**SYN:** variable data type

**EX:** a number(10);

**Block:**  Block is grouped code or set of statements, it is classified into two types.

1. Anonymous block
2. Sub programs
3. **Anonymous Blocks:**
4. Anonymous blocks are those blocks which we can’t store permanently in database once the user logout from the database these blocks are completely destroyed.
5. These blocks can’t have any prototype that is those blocks can be called as unnamed blocks.
6. **Subprograms:**
7. Sub programs are those blocks which we can store permanently in database so that, the user can access these blocks at a particular point of time.
8. These blocks refer with same proper name or specific name that is so that these blocks as named PL/SQL.

**Structure of anonymous blocks:**

Declare

Declaration of variables

Begin

Statement 1;

Statement 2;

.

.

Statement n;

Exception

Exception handling statements;

End;

PL/SQL anonymous block contains four sections those are declare, begin, exception and end. Here declare and exception sections are optional.

**Comments in PL/SQL:** -- it is single line comment in PL/SQL no multi line comments in PL/SQL.

**p) Write a PL/SQL block print ‘Sathya Technologies’**

begin

dbms\_output.put\_line('Sathya Technologies');

end;

**Note:** To display the output we can click options in menu bar and select server output. Here we select custom and select on. Then we will get output.

**P) Write a pl/sql program input any numbers and find out their sum.**

declare

a number:=&a;

b number:=&b;

c number;

begin

c:=a+b;

dbms\_output.put\_line('sum is'||c);

end;

**p) Write a PL/SQL block input two numbers and interchange them.**

declare

a number:=&a;

b number:=&b;

c number;

begin

c:=a;

a:=b;

b:=c;

dbms\_output.put\_line('value of a is'||a);

dbms\_output.put\_line('Value of b is'||b);

end;

(OR)

declare

a number:=&a;

b number:=&b;

begin

a:=a+b;

b:=a-b;

a:=a-b;

dbms\_output.put\_line('value of a is'||a);

dbms\_output.put\_line('Value of b is'||b);

end;

**P) Write a PL/SQL block to converts feets into inches.**

declare

a number:=&a;

begin

dbms\_output.put\_line('enter feet values'||a);

a:=a\*12;

dbms\_output.put\_line('inches for give value'||a);

end;

**Conditional statements in PL/SQL:**

**Syntax:**

If(condition) then

Statement 1;

Statemnet 2;

Statement n:

Else

Condition 1;

Condition 2;

Condition n;

End if;

This is the conditional statement in the PL/SQL. Which checks the condition either true or false by means of relational operator such as >, <, >=, <= et.c.,

If the condition is satisfied it executes the statement between if and else. If the condition is not satisfied then it will execute the statements after else.

**Note:** Every if condition should ends with end if statemet..

**P) Write a PL/SQL block input two numbers and find biggest one.**

declare

a number:=&a;

b number:=&b;

begin

if(a>b) then

dbms\_output.put\_line('a is big');

elsif(a=b) then

dbms\_output.put\_line('both are same');

else

dbms\_output.put\_line('b is big');

end if;

end;

**P) Write a PL/SQL block input any positive number and check it out even or odd?**

declare

a number:=&a;

begin

if(a =0) then

dbms\_output.put\_line(a||'is neither even nor odd');

elsif(a MOD 2 =0) then

dbms\_output.put\_line(a||'is even');

else

dbms\_output.put\_line(a||'is odd');

end if;

end;

**P) Write a PL/SQL block input any positive number and check it is even or odd? Without using mod function or mod operator.**

declare

n number:=&n;

begin

if(2\*floor(n/2)=n) then

dbms\_output.put\_line('n is even');

else

dbms\_output.put\_line(' n id odd');

end if;

end;

**P) Write a PL/SQL block input customer number, Customer name, Current balance, transaction amount and transaction code (d-deposit, w- with draw) calculate net balance.**

declare

cno number:=&cno;

cname varchar2(10):='&cname';

cb number:= &cb;

tamt number:=&tamt;

tc varchar2(1):='&tc';

nb number;

begin

if(tc='d') then

nb:=cb+tamt;

dbms\_output.put\_line('The net balance is' ||nb);

else

if(tc='w') then

nb:=cb-tamt;

dbms\_output.put\_line('The net balance is' ||nb);

else

dbms\_output.put\_line('Please enter d or w');

end if;

end if;

end;

**P) Write Pl/SQL block input consumer number, Consumer name, and starting reading number ending reading number, slap type (i-industry, c-commercial, r-residence). Calculate units consumed?**

**Conditions:**

1. **If slab type is industry then unit rate is 5/-**
2. **If slab type is commercial unit rate is 4/-**
3. **If slab type is residence unit price is 3/-. Calculate the total charge?**

declare

cno number(10):=&no;

cname varchar2(10):='&cname';

stno number(10):=&stno;

edno number(10):=&edno;

slab varchar2(1):='&slab';

tu number;

begin

tu:=edno+stno;

if(slab='i') then

tu:=tu\*5;

dbms\_output.put\_line('total unit cost is'||tu);

elsif(slab='c') then

tu:=tu\*4;

dbms\_output.put\_line('total unit cost is'||tu);

elsif(slab='r') then

tu:=tu\*3;

dbms\_output.put\_line('total unit cost is'||tu);

else

dbms\_output.put\_line('Please enter i or c or r');

end if;

end;

**(OR)**

declare

cno number(10):=&no;

cname varchar2(10):='&cname';

stno number(10):=&stno;

edno number(10):=&edno;

slab varchar2(1):='&slab';

tu number;

begin

tu:=edno+stno;

if(slab='i') then

tu:=tu\*5;

dbms\_output.put\_line('total unit cost is'||tu);

else

if(slab='c') then

tu:=tu\*4;

dbms\_output.put\_line('total unit cost is'||tu);

else

if(slab='r') then

tu:=tu\*3;

dbms\_output.put\_line('total unit cost is'||tu);

else

dbms\_output.put\_line('Please enter i or c or r');

end if;

end if

end if;

end;

**P) Write a PL/SQL block input consumer number, name, starting reading number, ending reading number and slab type(i-industry, c-commercial, r-residence) calculate units consumed.**

**Conditions:**

1. **If slab type is industry and units consumed <=100 then unit rate is 5/- else 6/-**
2. **If slab type is commercial and units consumed >=100 then unit rate is 4/- else5/-.**

**If slab type is residence and units consumed <=100 then unit rate is 3/- else 4/- calculate total charge.**

declare

cno number(10):=&no;

cname varchar2(10):='&cname';

stno number(10):=&stno;

edno number(10):=&edno;

slab varchar2(1):='&slab';

tu number;

begin

tu:=edno+stno;

if(slab='i') then

if(tu<=100) then

tu:=tu\*5;

dbms\_output.put\_line('total unit cost is'||tu);

else

tu:=tu\*6;

dbms\_output.put\_line('total unit cost is'||tu);

end if;

end if;

if(slab='c') then

if(tu<=100) then

tu:=tu\*4;

dbms\_output.put\_line('total unit cost is'||tu);

else

tu:=tu\*5;

dbms\_output.put\_line('total unit cost is'||tu);

end if;

end if;

if(slab='r') then

if(tu<=100) then

tu:=tu\*3;

dbms\_output.put\_line('total unit cost is'||tu);

else

tu:=tu\*4;

dbms\_output.put\_line('total unit cost is'||tu);

end if;

end if;

end;

**Loop:**

* 1. Loop is a mechanism which is supported by every programming language C, COBOL, e.t.c.,
  2. Whenever we need to execute a single or more than statements **repeatedly** then we use loops.
  3. Loops are categorized into two types.

1. Range based loops
2. Condition based loops.
3. **Range based loops:** Range based loop is a loop statements which executes statements as long as initial value reaches the final value. Once control crossed the final value then automatically comes under the loop.

**EX:** For loop, for reverse.

1. **Control based loops:** condition based loop is a loop statements which executes the statements as long as the given condition is satisfied. Once the condition is not satisfied control automatically outer the loop.

**EX:** while, loop.

Note: In oracle increment/ decrement is not programmer responsibility.

**Syn:** for variable in initialization . . final value

Loop

Statement 1;

Statement 2;

Statement n;

**P) Print Sathya Technologies 10 times on console.**

declare

i number(10);

begin

for i in 1..10 loop

dbms\_output.put\_line('Sathya Technologies');

end loop;

end;

**(OR)**

declare

i number(10);

begin

for i in -5..4 loop

dbms\_output.put\_line('Sathya Technologies');

end loop;

end;

**Note:** If i is participating in loop it is not possible.

**P) To print 10 numbers on console.**

declare

i number;

begin

for i in 1..10 loop

dbms\_output.put\_line(i);

end loop;

end;

**P) To print 10 even numbers**

declare

i number;

begin

for i in 1..5 loop

dbms\_output.put\_line((i\*2));

end loop;

end;

**P) To print 10 odd numbers**

declare

i number;

begin

for i in 1..5 loop

dbms\_output.put\_line((i\*2)-1);

end loop;

end;

**P) To find factorial number.**

declare

i number;

n number:=&n;

f number:=1;

begin

for i in 1..n loop

f:=f\*i;

end loop;

dbms\_output.put\_line(f);

end;

**P) Check it is prime or not.**

declare

i number;

n number:=&n;

c number:=0;

begin

for i in 1..n loop

if(mod(n,i)=0) then

c:=c+1;

end if;

end loop;

if(c=2) then

dbms\_output.put\_line(n||'is prime');

else

dbms\_output.put\_line(n|| 'is not prime');

end if;

end;

**P) Enter any four digit number reverse it.**

declare

n number:='&n';

i number;

s number:=0;

r number;

begin

for i in 1..4 loop

r:=mod(n,10);

s:=s\*10+r;

n:=floor(n/10);

end loop;

dbms\_output.put\_line(s);

end;

**P) Give input four numbers and find sum of those numbers.**

declare

n number:='&n';

i number;

s number:=0;

r number;

begin

for i in 1..4 loop

r:=mod(n,10);

s:=s+r;

n:=floor(n/10);

end loop;

dbms\_output.put\_line(s);

end;

**P) Give input four numbers and find out their first and fourth number’s sum.**

declare

n number:='&n';

i number;

s number:=0;

r number;

begin

r:=mod(n,10);

for i in 1..3 loop

n:=floor(n/10);

end loop;

s:=n+r;

dbms\_output.put\_line(s);

end;

**P) Take a four digit number and find out middle numbers sum.**

declare

n number:=&n;

r number;

k number;

l number;

begin

n:=floor(n/10);

r:=mod(n,10);

n:=floor(n/10);

k:=mod(n,10);

l:=r+k;

dbms\_output.put\_line(l);

end;

**P) Find out the difference between all numbers of four digit number.**

declare

n number:=&n;

i number;

r number;

t number;

s number:=0;

d number;

begin

r:=mod(n,10);

n:=floor(n/10);

for i in 1..3 loop

t:=mod(n,10);

s:=s+t;

n:=floor(n/10);

end loop;

d:=r-s;

dbms\_output.put\_line(d);

end;

**P) Print the multiplication table.**

declare

n number:=&n;

i number;

begin

for i in 1..10 loop

dbms\_output.put\_line(n||'\*'||i||'='||n\*i);

end loop;

end;

**P) Write a PL/SQL block to print following triangle.**

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

declare

i number;

j number;

begin

for i in 1..5 loop

for j in 1..i loop

dbms\_output.put(j||' ');

end loop;

dbms\_output.put\_line(' ');

end loop;

end;

**OUTPUT:**

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

**Print Following \* triangle**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

declare

i number;

j number;

begin

for i in 1..10 loop

for j in 1..i loop

dbms\_output.put('\*'||' ');

end loop;

dbms\_output.put\_line(' ');

end loop;

end;

**OUTPUT:**

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

\* \* \* \* \* \*

\* \* \* \* \* \* \*

\* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \*

\* \* \* \* \* \* \* \* \* \*

declare

i number;

j number;

begin

for i in 1..10 loop

for j in 1..i loop

dbms\_output.put('Bhargav'||' ');

end loop;

dbms\_output.put\_line(' ');

end loop;

end;

SQL> /

Bhargav

Bhargav Bhargav

Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav

Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav Bhargav

PL/SQL procedure successfully completed.

**For Reverse:**

**SYN:**

For variable in reverse

Initial value .. final value loop

Statement 1;

Statement 2;

End loop;

**P) Write a PL/SQL block to print the numbers 10 to 1**

declare

i number;

begin

for i in reverse 1..10 loop

dbms\_output.put\_line(i);

end loop;

end;

**P) Write a PL/SQL block input any number and find factorial**

declare

i number;

f number:=1;

n number:=&n;

begin

for i in reverse 1..n loop

f:=f\*i;

end loop;

dbms\_output.put\_line(f);

end;

**While Loop:**

**Syn:**

While(condition) loop

Statement1;

.

.

Statement n;

End loop;

**P) write a Pl/SQL block to print 1 to 10 numbers.**

declare

i number:=1;

begin

while(i<=10) loop

dbms\_output.put\_line(i);

i:=i+1;

end loop;

end;

**P)Write a PL/SQL program to reverse a number**

declare

n number:=&n;

r number;

s number:=0;

begin

while(n>0) loop

r:= mod(n,10);

s:=s\*10+r;

n:=floor(n/10);

end loop;

dbms\_output.put\_line(s);

end;

**P) Write PL/SQL Block input any number and check it is palindrome or not.**

**Palindrome:** Palindrome means the number is same if read from reverse that is called palindrome.

declare

n number:=&n;

r number;

s number:=0;

m number;

begin

m:=n;

while(n>0) loop

r:= mod(n,10);

s:=s\*10+r;

n:=floor(n/10);

end loop;

if (m=s) then

dbms\_output.put\_line('Palindrome');

else

dbms\_output.put\_line('not palindrome');

end if;

end;

**P) Write a PL/SQL block print it is Armstrong or not.**

**Armstrong number:** Armstrong number means the sum of cubes of a number is same to that number.

**EX:** 153

declare

n number:=&n;

r number;

s number:=0;

m number;

begin

m:=n;

while(n>0) loop

r:= mod(n,10);

s:=s+power(r,3);

n:=floor(n/10);

end loop;

if (m=s) then

dbms\_output.put\_line('Armstrong');

else

dbms\_output.put\_line('not Armstrong');

end if;

end;

**P) Program to find the Strong number or not.**

**Strong:** Strong number means factorial of each number’s sum is equal to given number.

145=1!+4!+5!=1+24+120=145 This is Strong number.

declare

n number:=&n;

f number:=1;

r number;

s number:=0;

m number;

begin

m:=n;

while(n>0) loop

r:=mod(n,10);

f:=1;

while(r>0) loop

f:=f\*r;

r:=r-1;

end loop;

s:=s+f;

n:=floor(n/10);

end loop;

if(m=s) then

dbms\_output.put\_line('Strong number');

else

dbms\_output.put\_line('not Strong number');

end if;

end;

/

**Loop Syntax:**

**Syn:**

Loop

Statement 1;.

.

Statement n;

Exit when(condition)

End loop;

Here the statements are executed as long as the given condition is not satisfied control automatically comes outer loop.

**Write a PL/SQL block to print 10 times a String.**

declare

i number:=1;

begin

loop

dbms\_output.put\_line('Sathya Technologies');

i:=i+1;

exit when(i>10);

end loop;

end;

**EMBEDDED SQL:**

**Write a PL/SQL block print the employee details such as employee name, salary and deptno when employee number passes as in value.**

declare

a number;

b varchar2(10);

c number;

d number;

begin

select ename,sal,deptno into b,c,d from emp where empno=&a;

dbms\_output.put\_line(b||' ' ||c||' '||d);

end;

In the above example the programmer need to remember data types of the variables and its sizes and also need to take more than one variable to retrieve more than one column data. To avoid all the above said problems we need to use following attributes.

* 1. **%type:** This attribute is used to avoid data types of variables and also its sizes.

**EX:**

declare

a emp.empno%type;

b emp.ename%type;

c emp.sal%type;

d emp.deptno%type;

begin

select ename,sal,deptno into b,c,d from emp where empno=&a;

dbms\_output.put\_line(b||' '||c||' '||d);

end;

**%Row Type:** This attribute is used to avoid to mention data type and its sizes and also no need to take more than one variable.

**EX:**

declare

a emp%rowtype;

begin

select \* into a from emp where empno=&a;

dbms\_output.put\_line('empno' || a.empno);

dbms\_output.put\_line('ename' || a.ename);

dbms\_output.put\_line('sal' || a.sal);

end;

**P) Write a PL/SQL blok to hike the employee salaries 1000/- when deptno passed as in value.**

begin

update emp set sal=sal+1000 where deptno=&deptno;

end;

**P) Write a PL/SQL block insert the records into the employee table.**

declare

a emp.empno%type;

b emp.ename%type;

c emp.sal%type;

d emp.deptno%type;

begin

insert into emp(empno,ename,sal,deptno) values(&a,&b,&c,&d);

dbms\_output.put\_line('Values inserted');

end;

**P) Write a PL/SQL block delete the employee details empno passed as value.**

begin

delete emp where empno=&empno;

dbms\_output.put\_line('Records deleted sucessfully');

end;

**Note:** To delete and update no need to take any value.

**Exception Handling:** Errors are classified into two types those are

1. Syntactical errors
2. Logical errors.

**Syntactical errors:** These errors will be raise when the user violate the language rules, these errors will be raise at compile time.

**Logical Errors:** These errors will be raise if the program contains any logical mistakes and those will be raised at runtime. We can handle these errors by using the exception handling mechanism.

Exceptions are classified into two types those are

1. System defined ex caption
2. User defined exceptions.
3. **System defined exceptions:** An exception which is defined by the system internally those exceptions can be called as System defined exceptions.
4. **User defined Exceptions:** An exception which is defined by the user manually and programmatically those exceptions are called User defined exceptions.

**1. NO\_DATA\_FOUND EXCEPTION:** This exception will be raised when the user’s data not available in the table.

**P) Write a PL/SQL block to raise NO\_DATA\_FOUND exception.**

declare

a emp%rowtype;

begin

select \* into a from emp where empno=&a;

dbms\_output.put\_line('ename is'||a.ename);

dbms\_output.put\_line('salary is'||a.sal);

dbms\_output.put\_line('department no is'||a.deptno);

exception

when no\_data\_found then

dbms\_output.put\_line('Data is not available for given number plese enter another number');

end;

**P) Write a PL/SQL block to raise zero divide exception.**

declare

a number:=&a;

b number:=&b;

c number;

begin

c:=a/b;

dbms\_output.put\_line('division is'||c);

exception

when zero\_divide then

dbms\_output.put\_line('infinity');

end;

* 1. **Too many rows:** This exception will be raised when the user is trying to fetch more than one record at a time.

**EX:**

declare

a number;

b varchar2(10);

c number;

begin

select ename,sal into b,c from emp where deptno=&a;

dbms\_output.put\_line(b||' ' ||c);

end;

ERROR at line 1:

ORA-01422: exact fetch returns more than requested number of rows

ORA-06512: at line 6

**EX:2**

declare

a number;

b varchar2(10);

c number;

begin

select ename,sal into b,c from emp where deptno=&a;

dbms\_output.put\_line(b||' ' ||c);

exception

when too\_many\_rows then

dbms\_output.put\_line('It is not posible to fetch more than one row');

end;

**OUTPUT:**

Enter value for a: 30

old 6: select ename,sal into b,c from emp where deptno=&a;

new 6: select ename,sal into b,c from emp where deptno=30;

It is not posible to fetch more than one row

PL/SQL procedure successfully completed.

**Using more than two exceptions.**

declare

a emp%rowtype;

begin

select \* into a from emp where deptno=&a;

dbms\_output.put\_line('ename is'||a.ename);

dbms\_output.put\_line('salary is'||a.sal);

dbms\_output.put\_line('department no is'||a.deptno);

exception

when no\_data\_found then

dbms\_output.put\_line('Data is not available for given number plese enter another number');

When too\_many\_rows then

dbms\_output.put\_line('It is not posible to fetch more than one row');

end;

* 1. **Dup\_val\_on\_index:** This exception will be raised if the user trying to enter duplicate values under primary key constraint column or unique key constraint column.

**EX:**

declare

a emp.empno%type;

b emp.ename%type;

c emp.sal%type;

d emp.deptno%type;

begin

insert into emp(empno,ename,sal,deptno) values(&a,&b,&c,&d);

dbms\_output.put\_line('Values inserted');

exception

when dup\_val\_on\_index then

dbms\_output.put\_line('the entered number already in the table');

end;

* 1. **Value error:** This exception will be raised if the data types are not matching.

declare

a number;

b number;

c number;

d number;

begin

select ename,sal,deptno into b,c,d from emp where empno=&a;

dbms\_output.put\_line(b||' ' ||c||' '||d);

Exception

when value\_error then

dbms\_output.put\_line('type not mathced here');

end;

**OUTPUT:**

Enter value for a: 7788

old 7: select ename,sal,deptno into b,c,d from emp where empno=&a;

new 7: select ename,sal,deptno into b,c,d from emp where empno=7788;

type not mathced here

PL/SQL procedure successfully completed.

**User Defined Exceptions:**

declare

a number:=&a;

bhargav exception;

begin

if(a=10) then

dbms\_output.put\_line('it is positive');

end if;

if (a=0) then

raise bhargav;

end if;

exception

when bhargav then

dbms\_output.put\_line('a is nuetral');

end;

**Cursors:**

1. Cursor is a temporary SQL work area which is used to fetch more than one record from existing table.
2. Cursors are classified into two types those are
3. Implicit Cursors
4. Explicit Cursors.
5. **Implicit Cursors:** The cursor mechanism which is performed by the system internally those cursors can be called as implicit cursors.
6. **Explicit Cursors:** The Cursor mechanism which is performed by the user manually those cursors can be called as explicit cursors.

Whenever we are working with cursors with explicit cursors we need to perform following operations.

1. Declare Cursor
2. Open the cursor
3. Close the cursor
4. Fetch the records from the cursor.

**Syntax to Create a Cursor:**

CURSOR<Cursor Name> IS SELECT \* FROM <Table Name> WHERE <Condition>

**EX:** Cursor C is select \* from emp where deptno=20;

**Open the cursor:**

**Syn:** Open <Cursor name>

**EX:** Open c;

**Fetch the records from the Cursor:**

**SYN:**FETCH < Cursor Name> INTO <List of variables>

**EX:** Fetch C into x,y,z;

**Syntax to Close the Cursor:**

**Syn:** CLOSE <Cursor Name>

**EX:** Close C;

**A program which have no cursor.**

declare

a emp%rowtype;

begin

select \* into a from emp where deptno=&deptno;

dbms\_output.put\_line(a.ename||' '||a.sal||' '||a.deptno);

end;

ERROR at line 1:

ORA-01422: exact fetch returns more than requested number of rows

ORA-06512: at line 4

**To overcome this program**

**P) Write a PL/SQL Block to display employee details such as employee no, ename, salary when department no is passed as in value.**

declare

a emp%rowtype;

cursor c1 is select \* from emp where deptno=&deptno;

begin

open c1;

fetch c1 into a;

dbms\_output.put\_line(a.ename||' '||a.sal||' '||a.deptno);

close c1;

end;

The above example fetches only one record because the statements are not lies within the loop. Whenever we are working with loops we need to use the following attributes.

1. %Found
2. %Not Found
3. %Open
4. %RowCount
5. **%Found:**
6. This attribute is used to check whether the record is found or not in the memory it always retrurns Boolean values that is either true or false.
7. If the record is found then it is true if the recors is not found then it is false.

**P) Write a PL/SQL Block display the employee details who are working under Sales Department.**

declare

a emp%rowtype;

cursor c is select \* from emp where deptno=&deptno;

begin

open c;

loop

fetch c into a;

if c% found then

dbms\_output.put\_line(a.empno||' '||a.sal||' '||a.deptno);

else

exit;

end if;

end loop;

end;

**Note:** We can use Fetch command for empno also but it will display only one value.

**P) Write a PL/SQL block to display the employee and department details.**

declare

a emp%rowtype;

cursor c1 is select \* from emp where deptno=(select deptno from dept where dname='SALES');

begin

open c1;

loop

fetch c1 into a;

if c1% found then

dbms\_output.put\_line(a.empno||' '||a.ename||' '||a.sal);

else

exit;

end if;

end loop;

\* end;

**OUTPUT:**

7499 RAVI 5799

7521 ROHAN 5799

7654 PAVAN 5799

7698 VISHAL 1234

7844 SANJAY 1500

7900 GOPAL 5799

**EX: 2:**

declare

a emp%rowtype;

b dept%rowtype;

cursor c1 is select \* from emp where deptno=&deptno;

cursor c2 is select \* from dept where deptno=&deptno;

begin

open c1;

loop

fetch c1 into a;

if c1% found then

dbms\_output.put\_line(a.empno||' '||a.ename||' '||a.sal);

else

exit;

end if;

end loop;

open c2;

loop

fetch c2 into b;

if c2% found then

dbms\_output.put\_line(b.dname);

else

exit;

end if;

end loop;

end;

1. **%Isopen:** This attribute is used to check whether the cursor is opened or not in memory.

**EX:**

declare

a emp%rowtype;

cursor c1 is select \* from emp;

begin

if c1% isopen then

dbms\_output.put\_line('Cursor not opened');

else

open c1;

dbms\_output.put\_line('Cursor Opened');

end if;

end;

1. **%Not Found:** This attribute is used to check whether the record is found or not in the cursor. This attribute always say us Boolean value that is true or false if the record is found then it is false. If the record is not found then it is true.

**%Row Count:** This attribute is used to count the number of records with in the cursor.

**Reference Cursor:** It is special type of cursor. We can assign more than one select statement to same cursor.

declare

type curs\_bhar is ref cursor;

a emp%rowtype;

b dept%rowtype;

c curs\_bhar;

begin

open c for select \* from emp;

loop

fetch c into a;

if c% found then

dbms\_output.put\_line(a.empno||' '||a.ename||a.sal);

else

exit;

end if;

end loop;

open c for select \* from dept;

loop

fetch c into b;

if c% found then

dbms\_output.put\_line(b.dname);

else

exit;

end if;

end loop;

end;

**Advantages of for loops using in the Cursors:**

1. No need to open the Cursor.
2. Fetch the records automatically.
3. It automatically check the end of the rows.
4. It automatically close the Cursor.
5. No need to declare the variables.
6. Code will be decreased.
7. Execution will be faster.
8. It is collection of information from cursor to variable (That is i).

**P) Write a PL/SQL block using % not found attribute.**

declare

a emp%rowtype;

cursor c is select \* from emp;

begin

if c% isopen then

dbms\_output.put\_line('I think C not opened');

else

open c;

end if;

loop

fetch c into a;

dbms\_output.put\_line(a.empno||' '||a.ename||' '||a.sal||' '||a.deptno);

exit when c% notfound;

end loop;

dbms\_output.put\_line(c%rowcount||'rows are displayed');

end;

**For loop using the Cursor.**

**EX:**

declare

cursor c is select \* from emp;

begin

for i in c loop

dbms\_output.put\_line(i.deptno||' '||i.sal||' '||i.ename);

end loop;

end;

**P) Write a PL/SQL block display the employee details whose name contains letter A working in CHICAGO.**

declare

cursor c is select \* from emp where deptno =(select deptno from

dept where loc ='NEW YORK') and ename like '%A%';

begin

for i in c loop

dbms\_output.put\_line(i.empno||' '||i.ename||' '||i.sal);

end loop;

end;

**CURSOR\_ALREADY\_OPEN:** This exception will be raised if the user trying to open the cursor within the for loop program.

declare

cursor c is select \* from emp where deptno =(select deptno from

dept where loc ='NEW YORK') and ename like '%A%';

begin

for i in c loop

open c;

dbms\_output.put\_line(i.empno||' '||i.ename||' '||i.sal);

end loop;

exception

when cursor\_already\_open then

dbms\_output.put\_line('It is not necessary to open cursor in for loop');

end;

**Cursors using joins:**

declare

cursor c\_emp is

select ename,sal\*12 ANNSAL,dname

from emp,dept

where emp.deptno = dept.deptno;

begin

for i in c\_emp

loop

dbms\_output.put\_line(i.ename || ' - ' || i.annsal || ' - ' || i.dname);

end loop;

end;

**Parameterized Cursors:**

declare

cursor c(dno number) is select \* from dept where deptno=dno;

begin

for i in c(10) loop

dbms\_output.put\_line(i.deptno||' '||i.dname);

end loop;

end;

**EX: 2**

declare

cursor c(dno number) is select \* from dept where deptno=dno;

begin

for i in c(&dno) loop

dbms\_output.put\_line(i.deptno||' '||i.dname);

end loop;

end;

**EX:**

cursor c(dno number) is select \* from dept where deptno=dno;

begin

for i in c(&dno) loop

dbms\_output.put\_line(i.deptno||' '||i.dname);

end loop;

end;

**Implicit Cursors:**

**EX:**

declare

vdeptno number(3):=&vdeptno;

begin

update emp set sal=sal+1000 where deptno=vdeptno;

if sql% notfound then

dbms\_output.put\_line('Unable to update rows');

else

dbms\_output.put\_line(sql%rowcount||'emp details updated successfully');

end if;

end;

**EX2:**

declare

vdeptno number(5):=&vdeptno;

begin

delete from emp where deptno=vdeptno;

if sql% notfound then

dbms\_output.put\_line('Unable to update rows');

else

dbms\_output.put\_line(sql%rowcount||'emp details deletede sucessfully');

end if;

end;

**Note:** In implicit cursors we can use update and delete statements only. We can’t use select, insert.

**SUB PROGRAMS:**

1. Sub program is a part of the program which is used for code re usability.
2. Sub programs have two names those are procedures and functions.
3. The main difference between the procedure and the function is procedure does not return any value but function returns one value.

PL/SQL Procedures are two parts.

1. Declaration of the procedure
2. Body of the procedure

Declaration of the procedure always start with a keyword called create and ends with last variable parameter.

Whereas the body of the procedure starts with a keyword called as **Is** and ends with **End** statement.

In PL/SQL sub program has three modes

1. IN MODE
2. OUT MODE
3. IN/OUT MODE
4. **IN MODE:** It is a default mode which is used to read the variables form the end user.
5. **OUT MODE:** This mode is used for writing purpose it sends the value to the end user.
6. **IN/OUT:** This mode is used for both reading and writing purpose it accepts and sends the values to the end user.

**Example for Procedure:**

create or replace procedure p1

(a in number,

b out number) as

begin

select sal into b from emp where empno=a;

end;

**EX:**

create or replace procedure k

(a in number,

b out varchar,

c out number,

d out number)

as

begin

select ename,sal,deptno into b,c,d from emp where empno=a;

end;

Procedure created.

SQL> var x varchar2(15)

SQL> var y number

SQL> var z number

SQL> exec k(7788,:x,:y,:z)

PL/SQL procedure successfully completed.

SQL> print x

X

--------------------------------

RAJU

SQL> print y

Y

----------

5699

SQL> print z

Z

----------

10

SQL> print

X

--------------------------------

RAJU

Y

----------

5699

Z

----------

10

Empno ename sal deptno

a b c d

7788 x y z

**Note:** If we replace existing procedure type create or replace

**Note:** For getting errors type show errors.

**Note:** For getting a value as input and we just specify like this a in out number;

**Note:** Number of OUT Parameters in the procedure should be equal to number of host variables and data types of OUT parameters should matched with host variables data types.

**P) Write a PL/SQL procedure change the employee salaries who are working under 30th department**

create or replace procedure p1 is

begin

update emp set sal=5000 where deptno=10;

end;

**P) Write a Procedure to delete the record from the table.**

create or replace procedure p4

(a in emp.empno%type)

is

begin

delete from emp where empno=a;

end;

**P) Write a procedure to insert the records into the table.**

create or replace procedure p4

(a emp.empno%type,

b emp.ename%type,

c emp.sal%type,

d emp.deptno%type)

is

begin

insert into emp(empno,ename,sal,deptno) values (a,b,c,d);

dbms\_output.put\_line('Record inserted sucessfully');

end;

**Write a Program to display employee details by using %rowtype**

create or replace procedure p4

(a in emp.empno%type) as b emp%rowtype;

begin

select \* into b from emp where empno=a;

dbms\_output.put\_line(b.ename);

dbms\_output.put\_line(b.sal);

end;

**Procedures with Cursors:**

create or replace procedure p2(p\_deptno in emp.deptno%type)

is

p\_empno emp.empno%type;

p\_ename emp.ename%type;

p\_sal emp.sal%type;

cursor c is select empno,ename, sal from emp where deptno=p\_deptno;

begin

open c;

loop

fetch c into p\_empno,p\_ename,p\_sal;

if c% found then

dbms\_output.put\_line(p\_empno);

dbms\_output.put\_line(p\_ename);

dbms\_output.put\_line(p\_sal);

else

exit;

end if;

end loop;

end;

**Procedures using for loop Cursors**

create or replace procedure p1 is

cursor c is select \* from emp where deptno=deptno;

begin

for i in c loop

dbms\_output.put\_line(i.ename);

end loop;

end;

**Function:**

1. Function is a self contained block which performs specific task.
2. The main advantage of function is code reusability.
3. Function returns one value.
4. Functions are classified into two types those are
5. **System Defined Function:** A function which is defined by the system along with the software those functions can be called as system defined functions.

**EX:** Length()

1. **User Defined Functions:** A function which is defined by the user manually or programmatically those functions can be called as user defined functions.

**EX:** Addd()

* Functions are used to estimate or calculate any value. Function contains two sections.

1. Declaration of the function
2. Body of the function.
3. Declaration of the function always starts with a keyword called create and ends with return statement where as body of the function start with a keyword called is and ends with end statement.

**Syntax:** CREATE OR REPLACE FUNCTION <Function Name>

(List of formal parameters)

RETURN TYPE

IS/AS

BEGIN

STATEMENT1

‘

‘

‘

STATEMENT N;

RETURN VARIABLE

END;

**P) Write a function to add two numbers.**

create or replace function kin(a in number,

b in number)

return number

is

c number;

begin

C:=a+b;

return c;

end;

**Execution Process:**

var x number

exec :x:=kin(10,20)

**P) Write a function input any value and return factorial.**

create or replace function fac(n in number)

return number

is

f number:=1;

i number;

begin

for i in 1..n loop

f:=f\*i;

end loop;

return f;

end;

**Execution Process**

SQL> var x number

SQL> exec :x:=fac(10)

PL/SQL procedure successfully completed.

SQL> print x

X

----------

3628800

**P) Write a PL/SQL function find out the annual salary of the employee when empno passed as in parameter.**

create or replace function pic(eno in number)

return number

is

c number;

begin

select sal\*12 into c from emp where empno=eno;

return c;

end;

**NOTE:**

1. In the above program if we specify empno instead of eno then we will get compile time error.
2. We may specify ‘in’ or may not specify at the place of declaration.

**Execution Process**

SQL> var x number

SQL> exec :x:=pic(7788)

PL/SQL procedure successfully completed.

SQL> print x

X

----------

60000

SQL> select pic(7788) from dual;

PIC(7788)

----------

60000

**Write a function display the maximum salary employee name**

create or replace function fun

return varchar

is

c varchar2(10);

begin

select ename into c from emp where sal=(select max(sal) from emp);

return c;

end;

**Execution process**

SQL> var x varchar2(15)

SQL> exec :x:=fun()

PL/SQL procedure successfully completed.

SQL> print x

X

-----------

Raj

**P) Write a Function find out sum of all employees salaries working under 10 th department**

create or replace function myfun

return number

is

c number;

begin

select sum(sal) into c from emp where deptno=10;

return c;

end;

**Execution Process**

SQL> var x number

SQL> exec :x:=myfun()

PL/SQL procedure successfully completed.

SQL> print x

X

----------

24999

**Write a function find out the maximum salary employee total salary working in SALES and ACCOUNTING department.**

create or replace function myfun

return number

is

c number;

begin

select sal+nvl(comm,0) into c from emp where sal=(select max(sal) from emp where deptno in

(select deptno from dept where dname in('SALES','ACCOUNTING')));

return c;

end;

**Execution Process**

SQL> var x number

SQL> exec :x:=myfun()

PL/SQL procedure successfully completed.

SQL> print x

**Package:**

1. Package is a collection of sub programs which contains functions and procedures.
2. Package contains two sections
3. Package Specification
4. Package Body
5. Package specification and package body name should starts with create and ends with end statement.
6. Here package specification name and package body name should be same.
7. Package contains only functions or only procedures or combinations both also.

**P) Develop a package display the employee details when employee number passed as IN parameter and find out the annual salary.**

create or replace package p10

is procedure s10

( a in emp.empno%type,

b out emp.ename%type,

c out emp.sal%type);

function s11

(a in emp.empno%type)

return number;

end;

X

----------

9999

**NOTE:** We never use Cursors in Functions.

**NOTE:** Package body can’t exist without package specification.

**P) Develop a package employee details when department number passed as IN value.**

create or replace package sixpack

is

procedure p1

(p\_deptno in emp.deptno%type);

function myfun

(p\_deptno in emp.deptno%type)

return number;

end;

create or replace package body sixpack

is procedure p1

(p\_deptno in emp.deptno%type)

is

a emp%rowtype;

cursor c is select \* from emp where deptno=p\_deptno;

begin

open c;

loop

fetch c into a;

if c% found then

dbms\_output.put\_line(a.empno||' '||a.sal||' '||a.deptno);

else

exit;

end if;

end loop;

end p1;

function myfun

(p\_deptno in emp.deptno%type)

return number is

c number;

begin

select count(\*) into c from emp where deptno=p\_deptno;

return c;

end myfun;

end;

**Execution Process**

SQL> var a number

SQL> exec :a:=sixpack.myfun(30)

PL/SQL procedure successfully completed.

SQL> print a

A

----------

6