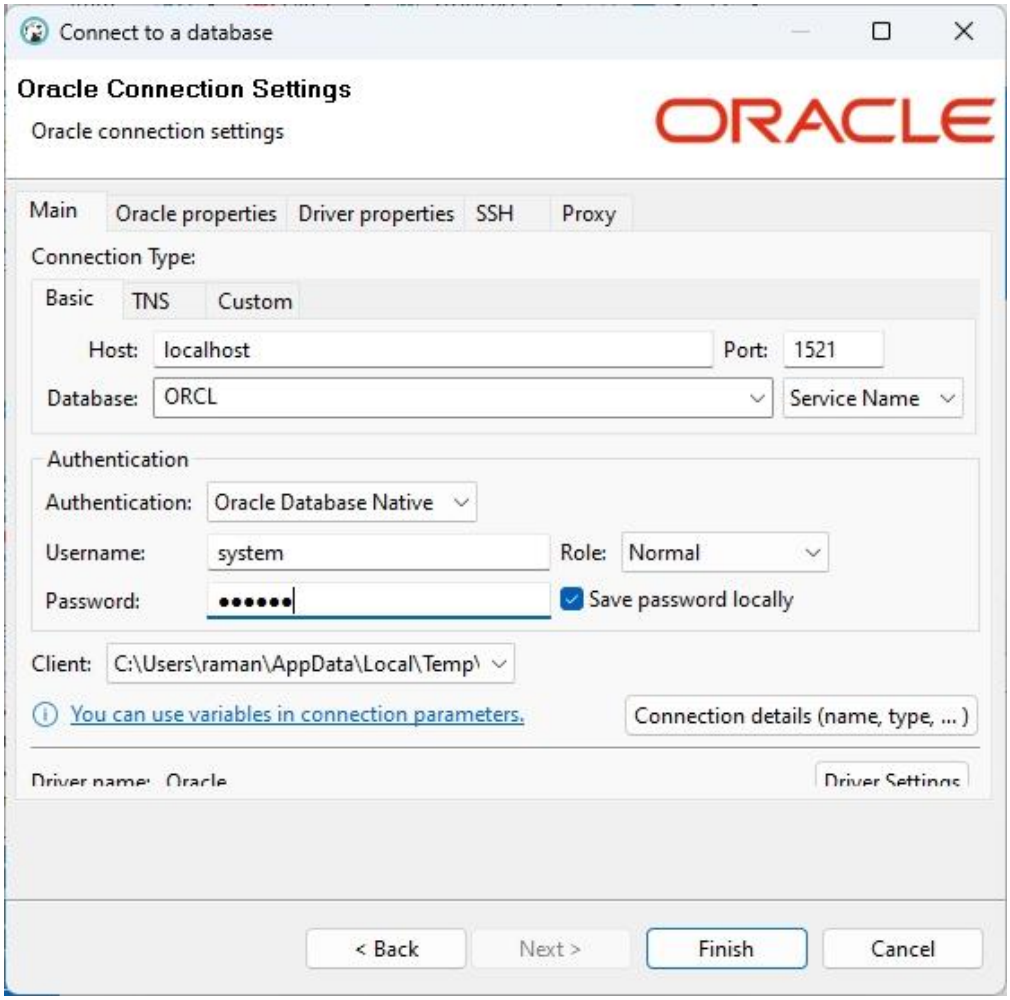
Oracle PL/SQL Handbook

Raman Deep Singh

We are using DBeaver for running pl/sql code in this article

you can also use SQL Developer for learning PL/SQL using this article

Here is the screenshot to connect to Oracle Schema



To use this article on PL/SQL

Make the following table in oracle schema you are using for this article

```
create table students(rollno number,name varchar2(20),marks number);
```

then insert five records in table students

```
insert into students values(1,'raman',90);

insert into students values(2,'aman',89);

insert into students values(3,'harman',78);

insert into students values(4,'ajay',66);

insert into students values(5,'vijay',77);
```

What is a View and How to create a view

View is a virtual table which means it does not physically exists. For example a view can be created to pull up few columns or records from table using a sql statement. Following is a view that pulls out only rollno and name from table students.

```
CREATE VIEW viewst AS SELECT rollno,name FROM students;
```

Following is the query to pull out records from viewst

```
SELECT * FROM viewst;
```

Blocks in PL/SQL

structure of a pl/sql program

declare (optional)

begin (mandatory)

exception (optional)

end; (mandatory)

---

example of anonymous block

declare

begin

null;

end;

---

there are three types of blocks

anonymous blocks

procedures

functions

---

pl/sql program to print hello world on output screen

set serveroutput on;

begin

dbms_output.put_line('HELLO WORLD');

end;

output

HELLO WORLD

---

set serveroutput on command will set serveroutput on if you will set serveroutput on the output will not be displayed on output window

---

program to declare a varchar2 variable and print the value in the variable

```
declare
    namestr varchar2(30);
begin
    namestr:='RAMAN';
    dbms_output.put_line(namestr);
end;
```

output

RAMAN

---

program to initialize a number variable and assign value 50 to it and print the value

```
declare
    a number(10);
begin
    a:=50;
    dbms_output.put_line(a);
end;
```

output

50

---

program to iniatize a number and assign value 50.24 to it and print the value

```
declare
    a number(5,2);
begin
    a:=50.24;
    dbms_output.put_line(a);
end;
```

output

50.24

---

program to find sum of two numbers

```
declare
```

```
    a NUMBER;
    b NUMBER;
   c NUMBER;
begin
    a:=10;
   b:=20;
  c:=a+b;
    dbms_output.put_line('Sum of a and b is ' || c);
end;
```

output

Sum of a and b is 30

---

program to find between two numbers

```
declare
    a NUMBER;
    b NUMBER;
   c NUMBER;
begin
    a:=100;
   b:=20;
  c:=a-b;
    dbms_output.put_line('Difference between a and b is ' || c);
end;
```

output

Difference between a and b is 80

---

program to multiply two numbers

```
declare
    a NUMBER;
    b NUMBER;
   c NUMBER;
begin
    a:=100;
   b:=20;
  c:=a*b;
    dbms_output.put_line('Product of a and b is ' || c);
end;
```

output

product of a and b is 2000

---

program to divide two numbers

```
declare
    a NUMBER;
    b NUMBER;
   c NUMBER;
begin
    a:=100;
   b:=20;
  c:=a/b;
    dbms_output.put_line('Quotient of a and b is ' || c);
end;
```

output

Quotient of a and b is 5

---

program to find remainder of two numbers

```
declare
     a NUMBER(10,2);
     b NUMBER(10,2);
    c NUMBER(10,2);
begin
     a:=101;
    b:=20;
   c:=a MOD b;
     dbms_output.put_line('Remainder of a and b is ' || c);
end;
```

output

Remainder of a and b is 1

---

program to print current date using date variable

```
DECLARE
       a DATE NOT NULL:= sysdate;
BEGIN
       dbms_output.put_line('Current Date is ' || a);
END;
```

output

Current Date is 10-JUN-23

---

program to print date and time with current timestamp and also to demonstrate timestamp datatype

```
DECLARE
       a TIMESTAMP NOT NULL:= systimestamp;
BEGIN
       dbms_output.put_line('Current Date with timestamp is ' || a);
END;
```

```
output

Current Date with timestamp is 10-JUN-23 10.11.46.372000 AM
```

---

program to print date and time and timezone

```
DECLARE
       a TIMESTAMP WITH time ZONE NOT NULL:= systimestamp;
BEGIN
       dbms_output.put_line('Current Date with timestamp is ' || a);
END;
```

output

```
Current Date with timestamp is 10-JUN-23 10.14.00.330000 AM +02:00
```

Program to demonstrate %type

%type means you are copying datatype of one variable to another

```
DECLARE
      a varchar2(20) :='RAMAN';
      b a%TYPE := 'AMAN';
BEGIN
      dbms_output.put_line('Value in a is ' || a);
      dbms_output.put_line('Value in b is ' || b);
END;
```

output

```
Value in a is RAMAN
Value in b is AMAN
```

Comments means line of codes which will not be executed they will be only read by the programmer.

Single line comments cover a single line and multiple line comments cover multiple lines.

Example of single line comment

```
DECLARE
      a varchar2(20) :='RAMAN';
BEGIN
      --This is a single line comment
      dbms_output.put_line('Value in a is ' || a);
END;
```

output

Value in a is RAMAN

Example of multiline comment

```
DECLARE
      a varchar2(20) :='RAMAN';
BEGIN
      /* This is a multiple

       line comment */
      dbms_output.put_line('Value in a is ' || a);
END;
```

Output

Value in a is RAMAN

Control Structures

if statement

if statement is used to check for a condition is true or not

```
DECLARE
	a NUMBER;
begin
	a:=10;
	IF a=10 then
	dbms_output.put_line('Value of a is equal to 10');
	ELSE
	dbms_output.put_line('Value of a is not equal to 10');
	END IF;
END;
```

output

Value of a is equal to 10

---

if statement to check whether number is greater than 10 or not

```
DECLARE
	a NUMBER;
begin
	a:=10;
	IF a>10 then
	dbms_output.put_line('Value of a is greater 10');
	ELSE
	dbms_output.put_line('Value of a is less than or equal to 10');
	END IF;
END;
```

```
output

Value of a is less than or equal to 10
```

---

if statement to check whether number is less than 10 or not

```
DECLARE
	a NUMBER;
begin
	a:=10;
	IF a<10 then
	dbms_output.put_line('Value of a is less than 10');
	ELSE
	dbms_output.put_line('Value of a is greater than or equal to 10');
	END IF;
END;
```

output

Value of a is greater than or equal to 10

---

if statement to check whether number is not equal to 10

```
DECLARE
	a NUMBER;
begin
	a:=10;
```

```
      IF a!=10 then
      dbms_output.put_line('Value of a is not equal to 10');
      ELSE
      dbms_output.put_line('Value of a is equal to 10');
      END IF;
END;
```

output

```
Value of a is not equal to 10
```

---

example of if elsif statement

if salary is between 1 and 20000 it will print salary is between 1 and 20000

if salary is between 20001 and 40000 it will print salary is between 200001 and 40000

```
DECLARE
      salary NUMBER;
begin
      salary:=25000;
      IF salary>0 AND salary<=20000 then
      dbms_output.put_line('Salary is between 1 and 20000');
      ELSIF salary>20000 AND salary<=40000 then
      dbms_output.put_line('Salary is between 20001 and 40000');
      else
      dbms_output.put_line('Salary is above 40000');
      END IF;
END;
```

output

```
Salary is between 20001 and 40000
```

Case Expressions

This is an example of case expression

if v_job_code= SA_MAN v_salary_increase will be 0.2

if v_job_code= SA_REP v_salary_increase will be 0.3

```
DECLARE
  v_job_code        VARCHAR2(10) := 'SA_MAN';
  v_salary_increase NUMBER;
BEGIN
  v_salary_increase :=  CASE v_job_code
                         WHEN 'SA_MAN' THEN 0.2
                         WHEN 'SA_REP' THEN 0.3
                        ELSE 0
                        END;
  dbms_output.put_line('Your salary increase is : '|| v_salary_increase);
END;
```

Output

```
Your salary increase is : .2
```

Example of CASE Statement

```
DECLARE
  v_job_code         VARCHAR2(10) := 'IT_PROG';
  v_department       VARCHAR2(10) := 'IT';
  v_salary_increase NUMBER;
BEGIN
  CASE
    WHEN v_job_code = 'SA_MAN' THEN
      v_salary_increase := 0.2;
      dbms_output.put_line('The salary increase for a Sales Manager is: '||
v_salary_increase);
    WHEN v_department = 'IT' AND v_job_code = 'IT_PROG' THEN
      v_salary_increase := 0.2;
      dbms_output.put_line('The salary increase for a Sales Manager is: '||
v_salary_increase);
    ELSE
      v_salary_increase := 0;
      dbms_output.put_line('The salary increase for this job code is: '||
v_salary_increase);
  END CASE;
END;
```

Output

```
The salary increase for a Sales Manager is: .2
```

Example of basic loop

```
DECLARE
  v_counter NUMBER(2) := 1;
BEGIN
  LOOP
    dbms_output.put_line('My counter is : '|| v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;
  END LOOP;
END;
```

Output

```
My counter is : 1
My counter is : 2
My counter is : 3
My counter is : 4
My counter is : 5
My counter is : 6
My counter is : 7
My counter is : 8
My counter is : 9
My counter is : 10
```

Example of while loop to print numbers from 1 to 10

```
DECLARE
  v_counter NUMBER(2) := 1;
BEGIN
  WHILE v_counter <= 10 LOOP
    dbms_output.put_line('My counter is : '|| v_counter);
    v_counter := v_counter + 1;
  END LOOP;
END;
```

Output

```
My counter is : 1
My counter is : 2
My counter is : 3
My counter is : 4
My counter is : 5
My counter is : 6
My counter is : 7
My counter is : 8
My counter is : 9
My counter is : 10
```

Example of for loop to print numbers from 1 to 10

```
BEGIN
  FOR i IN 1..10 LOOP
    dbms_output.put_line('My counter is : ' || i);
  END LOOP;
END;
```

```
My counter is : 1
My counter is : 2
My counter is : 3
My counter is : 4
My counter is : 5
My counter is : 6
My counter is : 7
My counter is : 8
My counter is : 9
My counter is : 10
```

Example of for loop to print numbers from 10 to 1 using REVERSE Keyword.

```
BEGIN
  FOR i IN REVERSE 1..10 LOOP
    dbms_output.put_line('My counter is : ' || i);
  END LOOP;
END;
```

```
My counter is : 10
My counter is : 9
My counter is : 8
My counter is : 7
My counter is : 6
My counter is : 5
My counter is : 4
My counter is : 3
My counter is : 2
My counter is : 1
```

Using SQL within PL/SQL
Following code can retrieve only 1 record from table

```
DECLARE
      vrollno NUMBER;
      vname   VARCHAR2(50);
      vmarks NUMBER;
BEGIN
  SELECT rollno,name,marks INTO vrollno,vname,vmarks FROM students WHERE
rollno=1;
  dbms_output.put_line('Rollno - ' || vrollno || ' Name - ' || vname || '
Marks - ' || vmarks);
END;
```

Above Code will give output

```
Rollno - 1 Name - raman Marks - 89
```

Example of sql Statement where we don't put a value in where condition but we use equality symbol in where condition

```
DECLARE
      vrollno NUMBER :=2;
      vname   VARCHAR2(50);
      vmarks NUMBER;
BEGIN
  SELECT rollno,name,marks INTO vrollno,vname,vmarks FROM students WHERE
rollno=vrollno;
  dbms_output.put_line('Rollno - ' || vrollno || ' Name - ' || vname || '
Marks - ' || vmarks);
END;
```

Output

```
Rollno - 2 Name - aman Marks - 84
```

Taking data from sql query using records or %type

vrollno students.rollno%**type**; means there is a table students and you are using datatype of column rollno as datatype of vrollno

```
DECLARE
      vrollno students.rollno%type;
      vname   students.name%TYPE;
      vmarks students.marks%TYPE;
BEGIN
  SELECT rollno,name,marks INTO vrollno,vname,vmarks FROM students WHERE
rollno=1;
  dbms_output.put_line('Rollno - ' || vrollno || ' Name - ' || vname || '
Marks - ' || vmarks);
END;
```

Output

```
Rollno - 1 Name - raman Marks - 89
```

Example to perform dml operations like insert into table students in pl/sql code

```
BEGIN
  INSERT INTO students values(6,'arman',88);
END;
```

What is a sequence?
sequence is used to increment a field or a column
For example you want to increase rollno by 1 whenever you want to insert record in table students.

You can create a sequence s1 which starts with 7 and increments by 1

```
CREATE SEQUENCE s1
START WITH 1
INCREMENT BY 1;
```

Now we will write pl/sql code to insert a record in table students using sequence s1
s1.nextval will give next number in the sequence

```
begin
      INSERT INTO students values(s1.nextval, 'ajay',79);
END;
```

---

currval will give current value of sequence

```
begin
dbms_output.put_line(s1.currval);
END;
```

Records in PL/SQL

Records means you can take whole row in table students in a record like s_record and you can declare a record using %ROWTYPE.
for example to create a record s_rec which matches a row in table students you can write

```
s_rec students%ROWTYPE;
```

Example of record

```
DECLARE
s_rec students%ROWTYPE;
BEGIN
      SELECT * INTO s_rec FROM students WHERE rollno=1;
      dbms_output.put_line('Rollno : ' || s_rec.rollno || ' Name : ' ||
s_rec.name || ' Marks : ' || s_rec.marks);
END;
```

Output

```
Rollno : 1 Name : ajay Marks : 79
```

Example of composite datatype varrays
varray is a fixed sized array.
array is a collection of records of fixed size.

```
DECLARE
TYPE names_list IS varray(5) OF varchar2(20);
names names_list;
BEGIN
      names:=names_list('raman','aman','harman','sunil','vijay');
FOR i IN 1..5 LOOP
      dbms_output.put_line(names(i));
END LOOP;

END;
```

Output

```
raman
aman
harman
sunil
vijay
```

---

count method or function of varrays

```
DECLARE
TYPE names_list IS varray(5) OF varchar2(20);
names names_list;
BEGIN
      names:=names_list('raman','aman','harman','sunil','vijay');
FOR i IN 1..names.count() LOOP
      dbms_output.put_line(names(i));
END LOOP;

END;
```

Output

```
raman
aman
harman
sunil
vijay
```

---

first and last method of varrays

```
DECLARE
TYPE names_list IS varray(5) OF varchar2(20);
names names_list;
BEGIN
      names:=names_list('raman','aman','harman','sunil','vijay');
FOR i IN names.FIRST() .. names.last() LOOP
      dbms_output.put_line(names(i));
END LOOP;

END;
```

Output

```
raman
aman
harman
sunil
vijay
```

---

example of exists function

```
DECLARE
TYPE names_list IS varray(5) OF varchar2(20);
names names_list;
BEGIN
      names:=names_list('raman','aman','harman','sunil','vijay');
FOR i IN 1 .. 5 LOOP
      IF names.exists(i) then
      dbms_output.put_line(names(i));
      END IF;
END LOOP;

END;
```

Output

```
raman
aman
harman
sunil
vijay
```

---

example of limit function

```
DECLARE
TYPE names_list IS varray(5) OF varchar2(20);
names names_list;
BEGIN
      names:=names_list('raman','aman','harman','sunil','vijay');
      dbms_output.put_line(names.limit());
END;
```

Output

```
5
```

Nested tables
Nested table is a composite datatype and are like varrays

example of nested table

```
DECLARE
TYPE names_list IS table OF varchar2(20);
names names_list;
BEGIN
      names:=names_list('raman','aman','harman','sunil','vijay');
      FOR i IN 1 .. names.count() loop
            dbms_output.put_line(names(i));
      END LOOP;
END;
```

Output

```
raman
aman
harman
sunil
vijay
```

Cursors
Explicit Cursors are used to pull records from sql table one by one.
Following is an example

```
DECLARE
   crollno students.rollno%type;
   cname students.name%type;
   cmarks students.marks%type;
   CURSOR c_student is
      SELECT rollno, name, marks FROM students;
BEGIN
   OPEN c_student;
   LOOP
   FETCH c_student into crollno, cname, cmarks;
      EXIT WHEN c_student%notfound;
      dbms_output.put_line('Rollno : ' || crollno || ' - Name : ' || cname
|| ' - Marks : ' || cmarks);
   END LOOP;
   CLOSE c_student;
END;
```

Output

```
Rollno : 2 - Name : aman - Marks : 89
Rollno : 3 - Name : harman - Marks : 77
Rollno : 1 - Name : raman - Marks : 90
```

Explanation of the above code

Above cursor will pull records from table students and place the values in variables crollno,cname and cmarks.
CURSOR Keyword declares a cursor.
open keyword opens a cursor.
loop keyword will run a loop and within the loop a cursor named as c_student will fetch records from table students one by one in variables crollno, cname and cmarks one by one and dbms_output.put_line will print the values line by line
end loop finishes the loop.
Close statement closes the cursor, it will close close the cursor when no more record is found.
`EXIT WHEN c_student%notfound;`
the above statement will check whether there are more records in the table or not.

Following cursor will pull records from table in a pl/sql record.

```
DECLARE
   csrecord students%rowtype;
   CURSOR c_student is
      SELECT rollno, name, marks FROM students;
BEGIN
   OPEN c_student;
   LOOP
   FETCH c_student into csrecord;
      EXIT WHEN c_student%notfound;
      dbms_output.put_line('Rollno : ' || csrecord.rollno || ' - Name : '
|| csrecord.name || ' - Marks : ' || csrecord.marks);
   END LOOP;
   CLOSE c_student;
```

```
END;
```

Output

```
Rollno : 2 - Name : aman - Marks : 89
Rollno : 3 - Name : harman - Marks : 77
Rollno : 1 - Name : raman - Marks : 90
```

What are Functions in PL/SQL

Functions mean is a group of statements which run as a whole and returns a value.

Following function will return maximum marks from table students

```
CREATE OR REPLACE FUNCTION findmaxmarks1
RETURN number
IS
fmarks number;
BEGIN
   SELECT max(students.marks) INTO fmarks FROM students.students;
  RETURN fmarks;
END findmaxmarks1;
```

Now How to call function findmaxmarks1

Following is the code

```
DECLARE
      fma number;
BEGIN
      fma:=findmaxmarks1();
      dbms_output.put_line('Maximum Marks in Table Students are : ' ||
fma);
END;
```

fma:=findmaxmarks1(); statement calls the function findmaxmarks1 and stores the returned value in fma variable.

Output

```
Maximum Marks in Table Students are : 90
```

Now we will create a function to return name of student through a rollno

```
CREATE OR REPLACE FUNCTION printname1(rno IN number)
RETURN students.NAME%type
IS
na students.NAME%type;
BEGIN
   SELECT name INTO na FROM students.students WHERE
students.students.rollno=rno;
  RETURN na;
END printname1;
```

Here the function name is printname1 rno is the parameter or argument passed to function
function returns a varchar2(20) variable because datatype of students.name column is varchar2(20).

Function Body starts after begin statement and declaration of variables is done after IS statement.
RETURN statement returns a varchar2(20) variable.

Now calling printname1 function

```
DECLARE
      nastr students.name%type;
BEGIN
      nastr:=printname1(1);
      dbms_output.put_line('Name in Table Students for Rollno 1 is : ' ||
nastr);
END;
```

Output

```
Name in Table Students for Rollno 1 is : raman
```

Procedures in PL/SQL is piece of code that is executed as a whole but it does not returns a value.

Following is the example

```
CREATE OR REPLACE procedure findmaxmarksp
IS
fmarks number;
BEGIN
   SELECT max(students.marks) INTO fmarks FROM students.students;
  dbms_output.put_line('Maximum Marks are : ' || fmarks);
END findmaxmarksp;
```

Following is the pl/sql block to call the procedure

```
BEGIN
      findmaxmarksp;
END;
```

Output

```
Maximum Marks are : 90
```

Following is the example of procedure in which we pass an argument to procedure

```
CREATE OR REPLACE procedure printnamep(rno IN number)
IS
na students.NAME%type;
BEGIN
   SELECT name INTO na FROM students.students WHERE
students.students.rollno=rno;
  dbms_output.put_line(na);
END printnamep;
```

Calling the above procedure

```
BEGIN
      printnamep(2);
END;
```

Output

aman

What are packages?

Packages are group of functions and packages in it

Following is a package that contains procedure hello which print hello

First of all we create a package with name as package4 and then create the package body. After that we call procedure hello by applying (.)

```
CREATE or replace PACKAGE package4 AS
   PROCEDURE hello;
END package4;
CREATE OR REPLACE PACKAGE BODY package4 AS
procedure hello
aS
BEGIN

dbms_output.put_line('Hello');
END;
end;
```

Following code will call procedure hello in package package4.

```
BEGIN
      package4.hello();

END;
```

Output

Hello

Following is an example of creating a function that will contain
a procedure hello and function hellof
hello will print hello
hellof will print Hello Function and will return value 0.

```
CREATE or REPLACE PACKAGE package5 AS
   PROCEDURE hello;
   function hellof return number;
END package5;
CREATE OR REPLACE PACKAGE BODY package5 AS
procedure hello
aS
BEGIN

dbms_output.put_line('Hello');
END;
function hellof return number
as
begin
dbms_output.put_line('Hello Function');
```

```
return 0;
END;
end;
/

DECLARE
a NUMBER;
BEGIN
      package5.hello();
      a:=package5.hellof();
END;
```

Output

```
Hello
Hello Function
```

Following is the procedure that will print records in table students using a cursor.

```
CREATE OR REPLACE PROCEDURE printcursor
AS
 crollno students.rollno%type;
   cname students.name%type;
   cmarks students.marks%type;
   CURSOR c_student is
      SELECT rollno, name, marks FROM students;
BEGIN
   OPEN c_student;
   LOOP
   FETCH c_student into crollno, cname, cmarks;
      EXIT WHEN c_student%notfound;
      dbms_output.put_line('Rollno : ' || crollno || ' - Name : ' || cname
|| ' - Marks : ' || cmarks);
   END LOOP;
   CLOSE c_student;
END;
```

Now call this procedure

```
begin
printcursor;
END;
```

Output

```
Rollno : 1 - Name : raman - Marks : 90
Rollno : 2 - Name : aman - Marks : 89
Rollno : 3 - Name : harman - Marks : 77
Rollno : 1 - Name : raman - Marks : 90
```

Triggers

Triggers are events or actions performed before or after any ddl or dml command is executed.

Following trigger will let user insert marks in table students if marks>=40 otherwise it will show error message that user cannot insert record.

```
CREATE OR REPLACE TRIGGER checkmarks1
```

```
BEFORE insert ON students.students
FOR each row
BEGIN
IF :NEW.marks<40 THEN
raise_application_error(-20007,'Marks cannot be less than 40');
END IF;
END;
```

When you will write query

```
insert into students values(1,'raman',63);
```

record will be inserted

when you will write query

```
insert into students values(1,'raman',33);
```

you will get following error

```
Error starting at line : 11 in command -

insert into students values(1,'raman',33)

Error at Command Line : 11 Column : 13

Error report -

SQL Error: ORA-20007: Marks cannot be less than 40

ORA-06512: at "STUDENTS.CHECKMARKS1", line 3

ORA-04088: error during execution of trigger 'STUDENTS.CHECKMARKS1'
```
