

7) Write a program to find the largest number from the given three numbers.

**declare**

**a number:=0;**

**b number:=0;**

**c number:=0;**

**begin**

**a:=&a;**

**b:=&b;**

**c:=&c;**

**if a>b and a>c then**

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

**elsif b>a and b>c then**

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

**else**

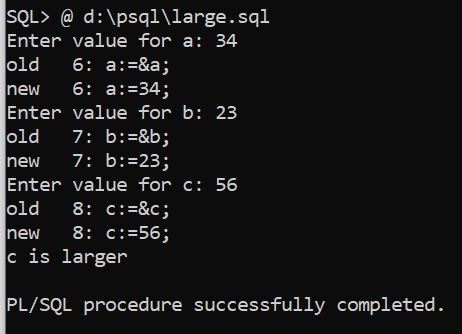
**dbms\_output.put\_line('c is larger');**

**end if;**

**end;**

**/**

**Output:-**



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

8) Simple programs using loops, while and for iterative control statements.

**a)--PL/SQL program to illustrate loop**

**declare**

**i number;**

**begin**

**i:=1;**

**loop**

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

**i:=i+1;**

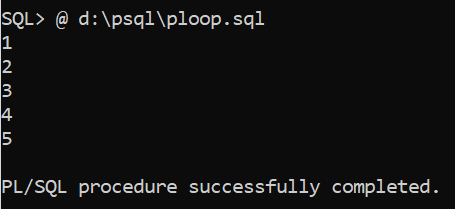
**exit when i>5;**

**end loop;**

**end;**

**/**

**Output:-**



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

**8b)Pl/sql program to illustrate while loop**

**declare**

**i number;**

**begin**

**i:=1;**

**while(i<=10)**

**loop**

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

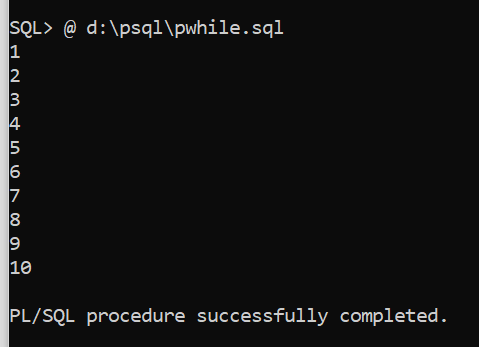
**i:=i+1;**

**end loop;**

**end;**

**/**

**Output:-**



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

**8c)Pl/Sql program to illustrate for loop.**

**declare**

**i number;**

**begin**

**for i in 1..15 loop**

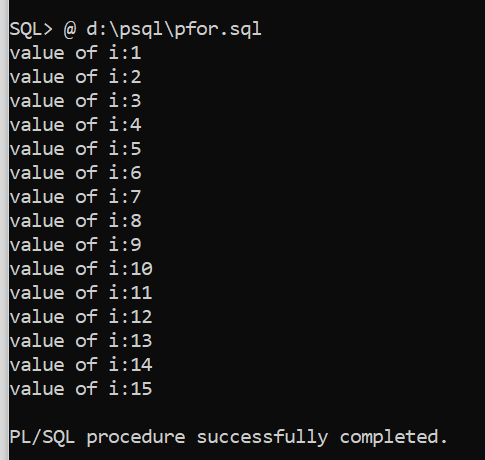
**dbms\_output.put\_line('value of i:'|| i);**

**end loop;**

**end;**

**/**

**Output:-**



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

9) Write a program to check whether the given number is Armstrong or not

**declare**

**n number;**

**c number;**

**m number;**

**r number;**

**begin**

**n:=&n;**

**c:=0;**

**m:=n;**

**while(n!=0)**

**loop**

**r:=mod(n,10);**

**c:=c+power(r,3);**

**n:=floor(n/10);**

**end loop;**

**if(m=c) then**

**dbms\_output.put\_line('number is an armstrong ');**

**else**

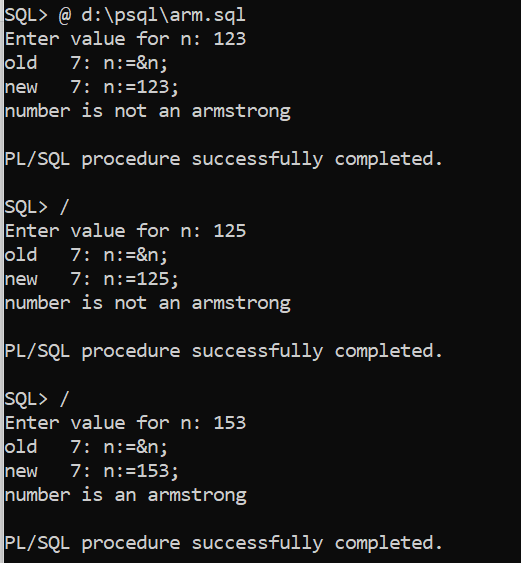
**dbms\_output.put\_line('number is not an armstrong');**

**end if;**

**end;**

**/**

**Output:-**



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

10) Write a program to generate all prime numbers below 100.

**Prime numbers:- prime numbers are the numbes which have only 2 factors 1 and the number itself**

**Eg:- 2,3,5,7,11,13,17,19,…………………**

**DECLARE**

**i NUMBER(3);**

**j NUMBER(3);**

**BEGIN**

**dbms\_output.Put\_line('The prime numbers are:');**

**dbms\_output.new\_line;**

**i := 2;**

**LOOP**

**j := 2;**

**LOOP**

**EXIT WHEN( ( MOD(i, j) = 0 ) OR ( j = i ) );**

**j := j + 1;**

**END LOOP;**

**IF( j = i )THEN**

**dbms\_output.Put(i||' ');**

**END IF;**

**i := i + 1;**

**exit WHEN i = 100;**

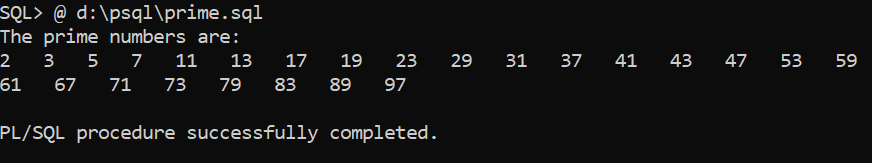
**END LOOP;**

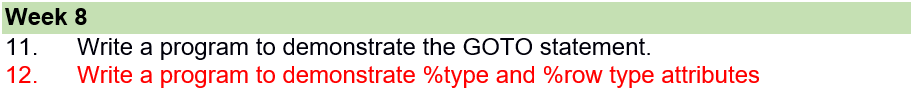
**dbms\_output.new\_line;**

**END;**

**/**

**Output:-**





11) Write a program to demonstrate the GOTO statement

The GOTO statement

This statement changes the flow of control within a PL/SQL Block. It allows the execution of a section of code which is not in the normal flow of control. The entry point into such a block of code is marked using the tags <<userdefined name>>. The GOTO statement can then make use of this user defined name to jump into the block of code for execution.

Syntax : GOTO <code-block name>

BEGIN

GOTO second\_message;

<<first\_message>>

DBMS\_OUTPUT.PUT\_LINE( 'Hello' );

GOTO the\_end;

<<second\_message>>

DBMS\_OUTPUT.PUT\_LINE( 'PL/SQL GOTO Demo' );

GOTO first\_message;

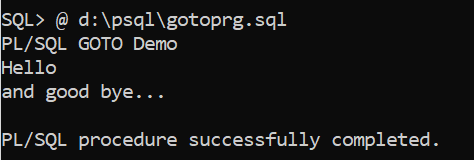
<<the\_end>>

DBMS\_OUTPUT.PUT\_LINE( 'and good bye...' );

END;

/

**Output:-**



/\* PL/SQL program to illustrate GO TO statement\*/

declare

i number;

begin

dbms\_output.put\_line('enter i value:');

i:=&i;

if(i>=0) then

goto here;

else

dbms\_output.put\_line(' i is negative');

end if;

goto abc;

<<here>>

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

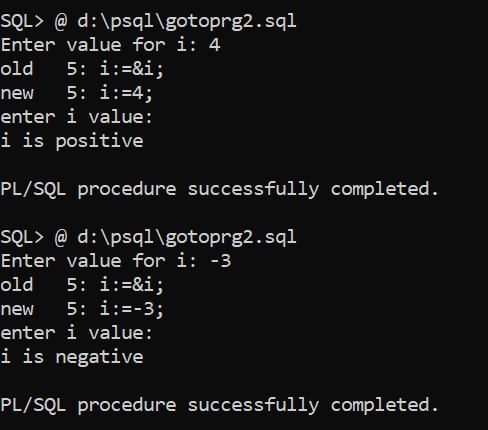
<<abc>>

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

end;

/

Output:-



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

12) Write a program to demonstrate %type and %row type attributes

Datatypes in SQL:-

1)Built in types

2)User defined data types

3)Reference type

Reference types:- 1) %type 2) %row type

%type:-

1)used to refer column type and size

2)used to match variable type and size with column type and size.

Eg:- vsal emp.sal%type;

3)What ever datatype &size declared for sal column the same type and size assigned to variable.

4)Advantage of %type is even if column type or size changes PL\SQL prg is not affected.

%ROWtype:-

1. Used to refer record type
2. A variable declared with %rowtype is called record type variable
3. In rowtype variable we can store row

Eg:- r emp%rowtype;

| Empno | Empname | Job | Manager | Hiredate | Sal | Comm | deptno |
| --- | --- | --- | --- | --- | --- | --- | --- |

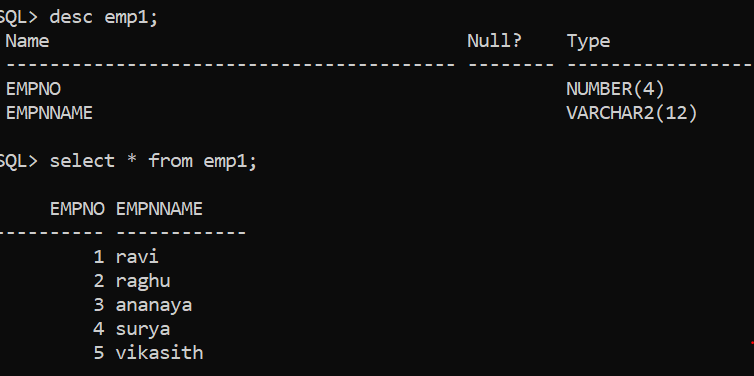
Select \* into r from emp where empno=7844;

r.sal=1500;

r.ename=tony

r.job=salesman

4)Advantage of %rowtype is it reduces number of variables required in the program and reduces complexity.



/\* illustration of %type and %row type\*/

declare

my\_empid emp1.empno%type;

my\_empname emp1.empnname%type;

my\_emprow emp1%rowtype;

no number:=0;

begin

no:=&no;

select empno,empnname into my\_empid,my\_empname from emp1 where empno=no;

if (sql%rowcount=1) then

dbms\_output.put\_line('empno is:' ||my\_empid|| ' , ' || 'empnname is : ' || my\_empname);

else

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

end if;

select \* into my\_emprow from emp1 where empno=no;

if (sql%rowcount=1) then

dbms\_output.put\_line('empno is:' ||my\_empid|| ' , ' || 'empname is :' ||my\_empname);

else

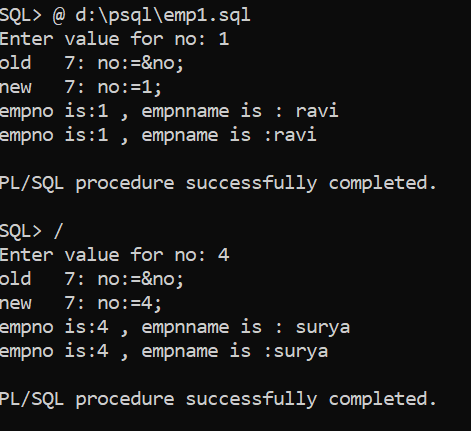
dbms\_output.put\_line('error');

end if;

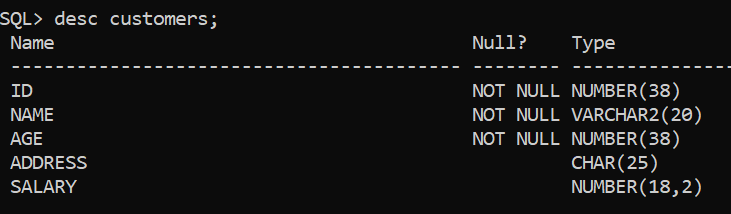
end;

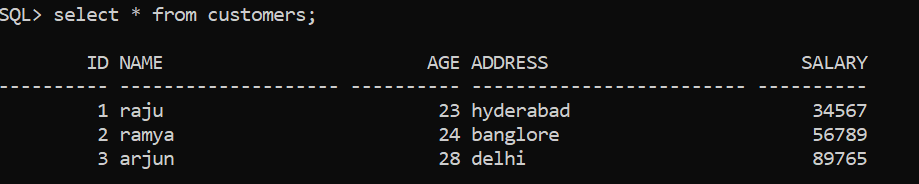
/

Output:-



Create customer table





DECLARE

cust Customers%rowtype;

BEGIN

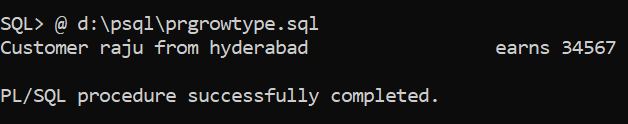
SELECT \* INTO cust FROM customers WHERE id = 1;

dbms\_output.put\_line ('Customer ' ||cust.name || ' from ' || cust.address || ' earns ' || cust.salary);

END;

/

**Output:-**



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

