**PL/SQL PROGRAMS**

**1. PL/SQL program to print "hai"**

set serveroutput on

begin

dbms\_output.Put\_line('hai');

end;

/

**hai**

PL/SQL procedure successfully completed

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**2.PL/SQL program to print a number and character.**

declare

var1 varchar(10);

num1 int(30);

begin

var1:='hello';

num1:=100;

dbms\_output.Put\_line('var1:'||var1);

dbms\_output.Put\_line('Num1:'||num1);

end;

/

var1:hello

Num1:100

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**3. PL/SQL program to retrieve data from table.**

create table Employees(eid int,fname varchar(30),salary int);

insert into Employees values(100,'ram',30000);

1 row created.

insert into Employees values(101,'sam',4000);

1 row created.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

declare

name varchar(10);

sal int(20);

begin

select fname,salary into name,sal from Employees where eid=100;

dbms\_output.Put\_line('Name:'||name);

dbms\_output.Put\_line('Salary:'||sal);

end;

/

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**4. PL/SQL program to retrieve data from table(same datatype from table)**

declare

name Employees.fname%TYPE;

sal Employees.salary%TYPE;

lname name%TYPE;

begin

select fname,salary into name,sal from Employees where eid=100;

dbms\_output.Put\_line('Name:'||name);

dbms\_output.Put\_line('Salary:'||sal);

end;

/

Name:ram

Salary:30000

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**5. PL/SQL program to find factorial of a givem number**

FACTORIAL OF A NUMBER

set serveroutput on;

declare

num number := 6;

fact number := 1;

temp number;

begin

temp :=num;

while( temp>0 )

loop

fact := fact\*temp;

temp := temp-1;

end loop;

dbms\_output.put\_line('factorial of '|| num || ' is ' || fact);

end;

/

factorial of 6 is 720

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**6. PL/SQL program to find factorial of any number.**

declare

n number;

fac number:=1;

i number;

begin

n:=&n;

for i in 1..n

loop

fac:=fac\*i;

end loop;

dbms\_output.put\_line('factorial='||fac);

end;

/

Enter value for n: 5

old 7: n:=&n;

new 7: n:=5;

factorial=120

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**7. PL/SQL program to add two given numbers**

ADDITION OF 2 NUMBERS

declare

x number(5);

y number(5);

z number(7);

begin

x:=10;

y:=20;

z:=x+y;

dbms\_output.put\_line('Sum is '||z);

end;

/

Sum is 30

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**8. PL/SQL program to add any two numbers**

Declare

Var1 integer;

Var2 integer;

Var3 integer;

Begin

Var1:=&var1;

Var2:=&var2;

Var3:=var1+var2;

Dbms\_output.put\_line(var3);

End;

/

Enter value for var1: 5

old 6: Var1:=&var1;

new 6: Var1:=5;

Enter value for var2: 6

old 7: Var2:=&var2;

new 7: Var2:=6;

11

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**9. PL/SQL program to perform arithmetic operations of given two numbers**

declare

num1 number;

num2 number;

num3 number;

begin

num1:=20;

num2:=15;

num3:=num1+num2;

dbms\_output.put\_line('The sum of '||num1||' and '||num2||' is '||num3);

num3:=num1-num2;

dbms\_output.put\_line('The subtraction of '||num1||' and '||num2||' is'||num3);

num3:=num1\*num2;

dbms\_output.put\_line('The multiplication of '||num1||' and '||num2||'is '||num3);

num3:=num1/num2;

dbms\_output.put\_line('The division of '||num1||' and '||num2||' is '||num3);

end;

/

The sum of 20 and 15 is 35

The subtraction of 20 and 15 is5

The multiplication of 20 and 15is 300

The division of 20 and 15 is 1.33333333333333333333333333333333333333

PL/SQL procedure successfully completed.

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

**10. PL/SQL PL/SQL program to display addition, subtraction, multiplication and division of two numbers and numbers will be entered by user.**

declare

num1 number;

num2 number;

num3 number;

begin

num1:=&num1;

num2:=&num2;

num3:=num1+num2;

dbms\_output.put\_line('The sum of '||num1||' and '||num2||' is '||num3);

num3:=num1-num2;

dbms\_output.put\_line('The subtraction of '||num1||' and '||num2||' is '||num3);

num3:=num1\*num2;

dbms\_output.put\_line('The multiplication of '||num1||' and '||num2||'is '||num3);

num3:=num1/num2;

dbms\_output.put\_line('The division of '||num1||' and '||num2||' is '||num3);

end;

/

Enter value for num1: 4

old 6: num1:=&num1;

new 6: num1:=4;

Enter value for num2: 5

old 7: num2:=&num2;

new 7: num2:=5;

The sum of 4 and 5 is 9

The subtraction of 4 and 5 is -1

The multiplication of 4 and 5is 20

The division of 4 and 5 is .8

PL/SQL procedure successfully completed.

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

**11.PL/SQL program to check whether entered number is even or odd using if-else statement.**

declare

x number;

begin

x:=&number;

if mod(x,2)=0 then

dbms\_output.put\_line('The number '||x||'is even');

else

dbms\_output.put\_line('The number '||x||'is odd');

end if;

end;

/

OUTPUT -

Enter value for number: 5

old 4: x:=&number;

new 4: x:=5;

The number 5 is odd

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

**12. PL/SQL program to find largest number of three numbers using if statement.**

declare

x number;

y number;

z number;

begin

x:=&number1;

y:=&number2;

z:=&number3;

if x>y and x>z then

dbms\_output.put\_line('x is largest');

end if;

if y>x and y>z then

dbms\_output.put\_line('y is largest');

end if;

if z>x and z>y then

dbms\_output.put\_line('y is largest');

end if;

end;

/

Enter value for number1: 4

old 6: x:=&number1;

new 6: x:=4;

Enter value for number2: 5

old 7: y:=&number2;

new 7: y:=5;

Enter value for number3: 1

old 8: z:=&number3;

new 8: z:=1;

y is largest

PL/SQL procedure successfully completed.

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

**13.PL/SQL program to find largest number of three numbers using if-elsif statement.**

declare

x number;

y number;

z number;

begin

x:=&number1;

y:=&number2;

z:=&number3;

if x>y and x>z then

dbms\_output.put\_line('x is largest');

elsif y>x and y>z then

dbms\_output.put\_line('y is largest');

else

dbms\_output.put\_line('z is largest');

end if;

end;

/

Enter value for number1: 7

old 7: x:=&number1;

new 7: x:=7;

Enter value for number2: 2

old 8: y:=&number2;

new 8: y:=2;

Enter value for number3: 3

old 9: z:=&number3;

new 9: z:=3;

x is largest

PL/SQL procedure successfully completed.

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

**PROCEDURE**

CREATE [OR REPLACE] PROCEDURE procedure\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

< procedure\_body >

END procedure\_name;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**14.PL/SQL procedure to print Hai**

set serveroutput on

CREATE OR REPLACE PROCEDURE welcome

AS

BEGIN

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

END;

/

Procedure created.

execute welcome

Hai!

**15.PL/SQL procedure to insert values into a table**

CREATE OR REPLACE PROCEDURE addemp

(depid IN dept.eid%TYPE,

depname IN dept.ename%TYPE,

depsal IN dept.psal%TYPE,

deploc IN dept.ploc%TYPE)

IS

BEGIN

INSERT INTO dept Values(depid,depname,depsal,deploc);

dbms\_output.Put\_line('dpmt added');

end;

/

procedure created

execute addemp(2,'MARY',3000,'klm');

dpmt added

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**16. To drop procedure**

PROCEDURE procedure-name;

DROP PROCEDURE welcome;

Procedure dropped.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**17.PL/SQL procedure to pfind minimum of two numbers.**

DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS

BEGIN

IF x < y THEN

z:= x;

ELSE

z:= y;

END IF;

END;

BEGIN

a:= 31;

b:= 56;

findMin(a, b, c);

dbms\_output.put\_line(' Minimum of (31,56: ' || c);

END;

/

Minimum of (31,56): 31

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**18.PL/SQL procedure to find square of a number.**

DECLARE

a number;

PROCEDURE squareNum(x IN OUT number) IS

BEGIN

x := x \* x;

END;

BEGIN

a:= 21;

squareNum(a);

dbms\_output.put\_line(' Square of (21): ' || a);

END;

/

Square of (21): 441

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**19.PL/SQL procedure to find sum of three numbers.**

DELIMITER //

CREATE PROCEDURE sumcalc()

BEGIN

DECLARE num1 INT;

DECLARE num2 INT;

DECLARE num3 INT;

DECLARE total INT;

SET num1 = 10;

SET num2 = 20;

SET num3 = 30;

SET total = num1 + num2 + num3;

SELECT total AS sum\_result;

END;

/

DELIMITER ;

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**20.PL/SQL procedure to retrieve student details from a table who have highest marks.**

set serveroutput on;

create or replace procedure topstud as topname student.name%type;

begin

select name into topname from student where marks=(select max(marks) from student);

dbms\_output.Put\_line(topname);

end;

/

Procedure created.

execute topstud;

sam

PL/SQL procedure successfully completed.

select \* from student;

ROLL NAME MARKS

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

1 ram 20

2 sam 30

3 nam 10

4 tam 25

**21.PL/SQL procedure to find factorial of a number**

DECLARE

num number;

factorial number;

FUNCTION fact(x number)

RETURN number

IS

f number;

BEGIN

IF x=0 THEN

f := 1;

ELSE

f := x \* fact(x-1);

END IF;

RETURN f;

END;

BEGIN

num:= 6;

factorial := fact(num);

dbms\_output.put\_line(' Factorial '|| num || ' is ' || factorial);

END;

/

factorial 6 is 720

PL/SQL procedure successfully completed.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**FUNCTIONS**

CREATE [OR REPLACE] FUNCTION function\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

RETURN return\_datatype

{IS | AS}

BEGIN

< function\_body >

END [function\_name];

Where,function-name specifies the name of the function.[OR REPLACE] option allows the modification of an existing function.

The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.The function must contain a return statement.

The RETURN clause specifies the data type you are going to return from the function.function-body contains the executable part.

The AS keyword is used instead of the IS keyword for creating a standalone function.

**22.PL/SQL function to find square of a number**

create function sqarea1(len int)return int is

area int(5,3);

begin

area:=(len\*len);

return area;

end;

/

Function created

begin

dbms\_output.Put\_line('area='||sqarea1(4));

end;

/

area=16

PL/SQL procedure successfully completed

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

**23.PL/SQL function to add two numbers**

create or replace function adder(n1 in number, n2 in number)

return number

is

n3 number(8);

begin

n3 :=n1+n2;

return n3;

end;

/

Function created.

DECLARE

n3 number(2);

BEGIN

n3 := adder(11,22);

dbms\_output.put\_line('Addition is: ' || n3);

END;

/

Addition is: 33

PL/SQL procedure successfully completed.

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

**24.PL/SQL function to find maximum of two numbers**

set serveroutput on

DECLARE

a number;

b number;

c number;

FUNCTION findMax(x IN number, y IN number)

RETURN number

IS

z number;

BEGIN

IF x > y THEN

z:= x;

ELSE

Z:= y;

END IF;

RETURN z;

END;

BEGIN

a:= 23;

b:= 45;

c := findMax(a, b);

dbms\_output.put\_line(' Maximum of (23,45): ' || c);

END;

/

Maximum of (23,45): 45

PL/SQL procedure successfully completed.

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

**25.PL/SQL function to find total numbers of cutomers in a table**

create table customers(id int,name varchar(20),dpmt varchar(20),salary int);

Table created.

insert into customers values(1,'sam','cs',2000);

1 row created.

insert into customers values(2,'tam','it',3000);

1 row created.

insert into customers values(3,'ram','ec',2500);

1 row created.

insert into customers values(4,'kam','mca',5000);

1 row created.

select \* from customers;

ID NAME DPMT SALARY

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

1 sam cs 2000

2 tam it 3000

3 ram ec 2500

4 kam mca 5000

CREATE OR REPLACE FUNCTION totalCustomers

RETURN number IS

total number(2) := 0;

BEGIN

SELECT count(\*) into total

FROM customers;

RETURN total;

END;

/

Function created.

DECLARE

c number(2);

BEGIN

c := totalCustomers();

dbms\_output.put\_line('Total no. of Customers: ' || c);

END;

/

Total no. of Customers: 4

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Cursor**- temporary area in main memory when a sql statement is executed.A cursor contains information on a select statement and the rows of data accessed by it.This temporary area is used to store the retrieved data from database and manipulate data.A cursor can hold more than one row,but can process only one row at a time.The set of rows the cursor holds is called active set.

Two types: implicit and explicit

**a) Implicit**: get created when you execute DMA queries .

The attributes to check status of DML operations:

%FOUND :if DML affects atleast one row returns true else return false

%NOTFOUND: if DML affects atleast one row returns false else true

%ROWCOUNT: returns number of rows affected by DML

|  |  |
| --- | --- |
| %ISOPEN | It always returns FALSE for implicit cursors, because the SQL cursor is automatically closed after executing its associated SQL statements. |

**26.PL/SQL program to update data in a table using cursor**

DECLARE

total\_rows number(2);

BEGIN

UPDATE customers

SET salary = salary + 5000;

IF sql%notfound THEN

dbms\_output.put\_line('no customers updated');

ELSIF sql%found THEN

total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' customers updated ');

END IF;

END;

/

4 customers updated

PL/SQL procedure successfully completed.

**b) Explicit:**

CURSOR cursor\_name IS select\_statement;;

* Declare cursor to initialize in the memory----CURSOR name IS  SELECT statement;
* Open the cursor to allocate memory------ OPEN cursor\_name;
* Fetch the cursor to retrieve data.-------- FETCH cursor\_name INTO variable\_list;
* Close the cursor to release allocated memory.------- Close cursor\_name;

**27.PL/SQL program to retreive id,name and salary from table using cursor**.

DECLARE

c\_id customers.id%type;

c\_name customers.name%type;

c\_salary customers.salary%type;

CURSOR c\_customers is

SELECT id, name, salary FROM customers;

BEGIN

OPEN c\_customers;

LOOP

FETCH c\_customers into c\_id, c\_name, c\_salary;

EXIT WHEN c\_customers%notfound;

dbms\_output.put\_line(c\_id || ' ' || c\_name || ' ' || c\_salary);

END LOOP;

CLOSE c\_customers;

END;

/

1 sam 7000

2 tam 8000

3 ram 7500

4 kam 10000

PL/SQL procedure successfully completed.

**Trigger**

Trigger is invoked by Oracle engine automatically whenever a specified event occurs.Trigger is stored into database and invoked repeatedly, when specific condition match.Triggers are stored programs, which are automatically executed or fired when some event occurs.Triggersare written to be executed in response to any of the following events.A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).A database definition (DDL) statement (CREATE, ALTER, or DROP).A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).Triggers could be defined on the table, view, schema, or database with which the event is associated.

Advantages of Triggers:

* Trigger generates some derived column values automatically
* Enforces referential integrity
* Event logging and storing information on table access
* Auditing
* Synchronous replication of tables
* Imposing security authorizations
* Preventing invalid transactions

SYNTAX

CREATE [OR REPLACE ] TRIGGER trigger\_name

{BEFORE | AFTER | INSTEAD OF }

{INSERT [OR] | UPDATE [OR] | DELETE}

[OF col\_name]

ON table\_name

[REFERENCING OLD AS o NEW AS n]

[FOR EACH ROW]

WHEN (condition)

DECLARE

Declaration-statements

BEGIN

Executable-statements

EXCEPTION

Exception-handling-statements

END;

* CREATE [OR REPLACE] TRIGGER trigger\_name: It creates or replaces an existing trigger with the trigger\_name.
* {BEFORE | AFTER | INSTEAD OF} : This specifies when the trigger would be executed. The INSTEAD OF clause is used for creating trigger on a view.
* {INSERT [OR] | UPDATE [OR] | DELETE}: This specifies the DML operation.
* [OF col\_name]: This specifies the column name that would be updated.
* [ON table\_name]: This specifies the name of the table associated with the trigger.
* [REFERENCING OLD AS o NEW AS n]: This allows you to refer new and old values for various DML statements, like INSERT, UPDATE, and DELETE.
* [FOR EACH ROW]: This specifies a row level trigger, i.e., the trigger would be executed for each row being affected. Otherwise the trigger will execute just once when the SQL statement is executed, which is called a table level trigger.
* WHEN (condition): This provides a condition for rows for which the trigger would fire. This clause is valid only for row level triggers

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

**28.PL/SQL program to create a row level trigger for the CUSTOMERS table that would fire for INSERT or UPDATE or DELETE operations performed on the CUSTOMERS table. This trigger will display the salary difference between the old values and new value.**

CREATE OR REPLACE TRIGGER salaryupdate

BEFORE DELETE OR INSERT OR UPDATE ON customers

FOR EACH ROW

WHEN (NEW.ID > 0)

DECLARE

sal\_diff number;

BEGIN

sal\_diff := :NEW.salary - :OLD.salary;

dbms\_output.put\_line('Old salary: ' || :OLD.salary);

dbms\_output.put\_line('New salary: ' || :NEW.salary);

dbms\_output.put\_line('Salary difference: ' || sal\_diff);

END;

/

Trigger created.

DECLARE

total\_rows number(2);

BEGIN

UPDATE customers

SET salary = salary + 5000;

IF sql%notfound THEN

dbms\_output.put\_line('no customers updated');

ELSIF sql%found THEN

total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' customers updated ');

END IF;

END;

/

Updating salary

Old salary: 7000

New salary: 12000

Salary difference: 5000

Old salary: 8000

New salary: 13000

Salary difference: 5000

Old salary: 7500

New salary: 12500

Salary difference: 5000

Old salary: 10000

New salary: 15000

Salary difference: 5000

4 customers updated

PL/SQL procedure successfully completed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**29.PL/SQL program to implement trigger to check salary is greater than 2000**

CREATE OR REPLACE TRIGGER Checksalary

BEFORE

INSERT OR UPDATE ON employee

FOR EACH ROW

BEGIN

IF :new.salary>3000 THEN

raise\_application\_error(-20001, 'salary should not be greater than 2000');

END IF;

END;

/

Trigger created.

update employee set salary=4000 where employee\_no=2;

update employee set salary=4000 where employee\_no=2

\*ERROR at line 1:

ORA-20001: Age should not be greater than 30

ORA-06512: at "SYSTEM.CHECKAGE", line 3

ORA-04088: error during execution of trigger 'SYSTEM.CHECKAGE'

update employee set salary=2600 where employee\_no=2;

1 row updated.

**29.PL/SQL program to implement trigger after inserting a record into table.**

create table stud(sid int,sname varchar(20),sdpmt varchar(20),marks int);

Table created.

insert into stud values(1,'sam','cse',20);

1 row created.

insert into stud values(2,'tam','it',30);

1 row created.

select \* from stud;

SID SNAME SDPMT MARKS

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

1 sam cse 20

2 tam it 30

create or replace trigger studtrig after insert on stud

begin

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

end;

/

Trigger created.

insert into stud values(3,'wam','ec',39);

inserted

1 row created.