**EXP NO : 11**

**230701329**

|  |  |  |
| --- | --- | --- |
| **Ex.No.: 11** | | **PL SQL PROGRAMS** |
| **Date:** | **26/10/24** |

# PROGRAM 1

Write a PL/SQL block to calculate the incentive of an employee whose ID is 110.

declare

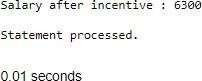
a employees.employee\_id%type; b employees.salary%type;

begin

Select salary into a from employees where employee\_id = 110; b:=0.05\*a;

dbms\_output.put\_line('Salary after incentive : '||(a+b)); end;

block to



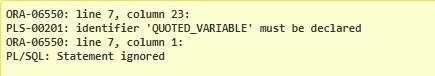
# PROGRAM 2

Write a PL/SQL show an invalid case-insensitive reference to a quoted and without quoted user-defined identifier.

declare

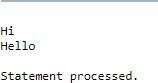
non\_quoted\_variable varchar2(10) := 'Hi'; "quoted\_variable" varchar2(10) := 'Hello'; begin

dbms\_output.put\_line(NON\_QUOTED\_VARIABLE); dbms\_output.put\_line("quoted\_variable"); dbms\_output.put\_line("QUOTED\_VARIABLE"); end;



# PROGRAM 3

Write a PL/SQL block to



adjust the salary of the employee whose ID

122. Sample table: employees

declare

old\_salary

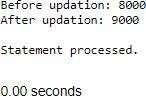
employees.salary%type; new\_salary

employees.salary%type; begin

new\_salary:= :sal;

Select salary into old\_salary from employees where employee\_id = 122; dbms\_output.put\_line('Before updation: '||old\_salary);

Update employees set salary = salary + new\_salary where employee\_id = 122; Select salary into new\_salary from employees where employee\_id = 122; dbms\_output.put\_line('After updation: '||new\_salary); end;



PROGRAM 4

Write a PL/SQL create a procedure using the "IS [NOT] NULL Operator" and show AND operator returns TRUE if and only if both operands are TRUE.

Create or replace procedure proc1( a boolean, b boolean) IS BEGIN

if(a is not null) and (b is not null) then if(a = TRUE and b = TRUE) then dbms\_output.put\_line('TRUE');

else dbms\_output.put\_line('FALSE'); end if;

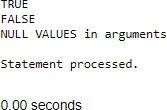
else

dbms\_output.put\_line('NULL VALUES in arguments'); end if;

end proc1;

BEGIN proc1(TRUE,TRUE); proc1(TRUE,FALSE); proc1(NULL,NULL);

end;



describe the usage of LIKE operator including wildcard characters

and escape character.

Declare

name varchar2(20); num number(3);

Begin num := :n;

Select first\_name into name from employees where employee\_id=num; if name like 'D%' then

dbms\_output.put\_line('Name starts with "D"'); end if;

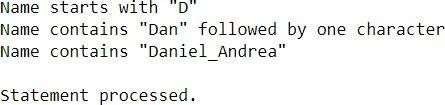
if name like 'Dan\_el%' then

dbms\_output.put\_line('Name contains "Dan" followed by one character'); end if;

name := 'Daniel\_Andrea';

if name like 'Daniel\\_Andrea' escape '\' then dbms\_output.put\_line('Name contains "Daniel\_Andrea"'); end if;

end;



PROGRAM 6

Write a PL/SQL program to arrange the number of two variable in such a way that the small number will store in num\_small variable and large number will store in num\_large variable.

declare

a number(2); b number(2);

num\_small number(2); num\_large number(2); begin

a := :s;

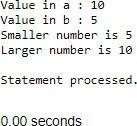
b := :l;

dbms\_output.put\_line('Value in a : '||a); dbms\_output.put\_line('Value in b : '||b); if a>b then

num\_small := b; num\_large := a; else

num\_small :=a; num\_large :=b; end if;

dbms\_output.put\_line('Smaller number is '||num\_small); dbms\_output.put\_line('Larger number is '||num\_large); end;



procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

Create or replace procedure calc\_incen(emp\_id number,achievement number,target number) AS

incentive number; rowcount number;

Begin

if achievement > target then incentive:= achievement\*0.2; else

incentive:=0; end if;

Update employees set salary = salary + incentive where employee\_id = emp\_id; rowcount:= SQL%ROWCOUNT;

if rowcount>0 then dbms\_output.put\_line('Record(s) updated'); else

dbms\_output.put\_line('No Record(s) updated'); end if;

end;

Declare

id number; achievement number; target number;

Begin

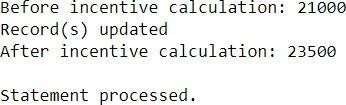
id := :emp\_id; achievement := :achieve; target := :target\_;

calc\_incen(id,achievement,target); end;



PROGRAM 8

Write a procedure to calculate incentive achieved according to the specific sale limit.



PROGRAM 9

Write a PL/SQL

Create or replace procedure calc\_incen(emp\_id number,sales number) AS incentive number;

rowcount number;

Begin

if sales < 1000 then incentive:= 0;

elsif sales > 1000 and sales < 2000 then incentive := sales \* 0.2;

else

incentive := sales \* 0.5; end if;

Update employees set salary = salary + incentive where employee\_id = emp\_id; rowcount:= SQL%ROWCOUNT;

if rowcount>0 then dbms\_output.put\_line('Record(s) updated'); else

dbms\_output.put\_line('No Record(s) updated'); end if;

end;

Declare

id number; sales number; sal number;

Begin

id := :emp\_id; sales := :sale;

select salary into sal from employees where employee\_id = id; dbms\_output.put\_line('Before incentive calculation: '||sal); calc\_incen(id,sales);

select salary into sal from employees where employee\_id = id; dbms\_output.put\_line('After incentive calculation: '||sal); end;

Write a PL/SQL to

program count number of employees in department 50 and check whether this department have any vacancies or not. There are 45 vacancies in this department.

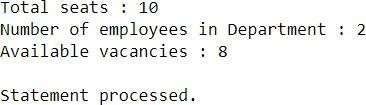
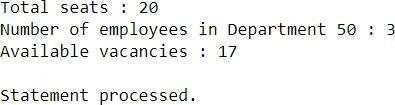
declare emp\_count number; vacancy number := 20;

begin

Select count(\*) into emp\_count from employees where department\_id = 10; dbms\_output.put\_line('Total seats : '||vacancy);

dbms\_output.put\_line('Number of employees in Department 50 : '||emp\_count); if emp\_count>vacancy then

dbms\_output.put\_line('No vacancies available'); else dbms\_output.put\_line('Available vacancies : '||(vacancy-emp\_count)); end if; end;



Write a PL/SQL program to count number of employees in a specific department and check whether this department have any vacancies or not. If any vacancies, how many vacancies are in that department.

Write a PL/SQL to

declare

dept\_id number; emp\_count number; vacancy number := 10; begin

dept\_id := :id;

Select count(\*) into emp\_count from employees where department\_id = dept\_id; dbms\_output.put\_line('Total seats : '||vacancy);

dbms\_output.put\_line('Number of employees in Department : '||emp\_count); if emp\_count>vacancy then

dbms\_output.put\_line('No vacancies available'); else

dbms\_output.put\_line('Available vacancies : '||(vacancy-emp\_count)); end if;

end;

program display the employee IDs, names, job titles, hire dates, and salaries of all employees.

begin

for i in (select employee\_id, first\_name, job\_id, hire\_date, salary from employees) loop

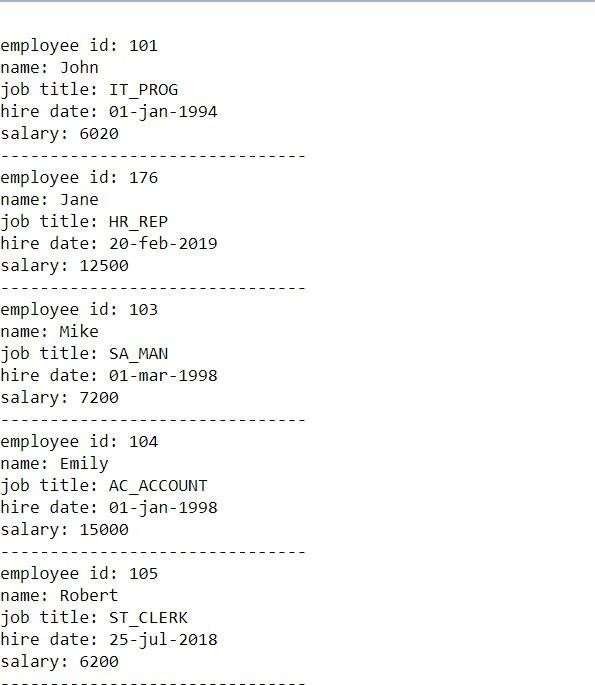
dbms\_output.put\_line('employee id: ' || i.employee\_id); dbms\_output.put\_line('name: ' || i.first\_name); dbms\_output.put\_line('job title: ' || i.job\_id);

dbms\_output.put\_line('hire date: ' || to\_char(i.hire\_date, 'dd-mon-yyyy')); dbms\_output.put\_line('salary: ' || i.salary);

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

end loop;

end;



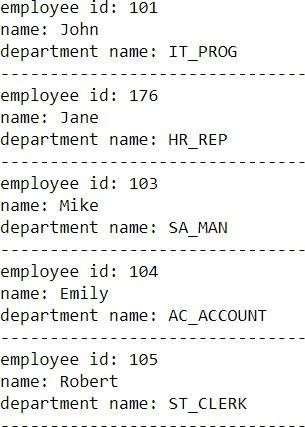
to

Write a PL/SQL program display the employee IDs, names, and department names of all employees.

begin

for i in (select e.employee\_id, e.first\_name, e.job\_id from employees e) loop dbms\_output.put\_line('employee id: ' || i.employee\_id); dbms\_output.put\_line('name: ' || i.first\_name); dbms\_output.put\_line('department name: ' || i.job\_id); dbms\_output.put\_line(' ');

end loop; end;



PROGRAM 13

Write a PL/SQL program display the job IDs, titles, and minimum salaries of all jobs.

Begin

for i in (select job\_id,job\_title,min\_salary from jobs) loop

dbms\_output.put\_line('job id: ' || i.job\_id); dbms\_output.put\_line('job title: ' || i.job\_title); dbms\_output.put\_line('minimum salary: ' || i.min\_salary); dbms\_output.put\_line(' ');

end loop; end;

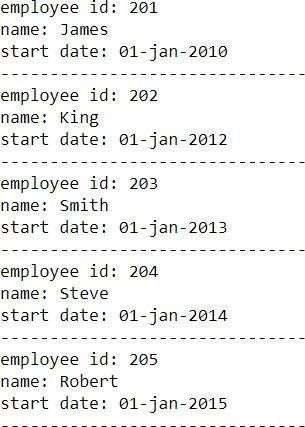


PROGRAM 14

Write a PL/SQL program display the employee IDs, names, and job history start dates of all employees.

Begin

for i in (select employee\_id,employee\_name,start\_date from job\_history) loop dbms\_output.put\_line('employee id: ' || i.employee\_id); dbms\_output.put\_line('name: ' || i.employee\_name); dbms\_output.put\_line('start date: ' ||to\_char(i.start\_date, 'dd-mon-yyyy')); dbms\_output.put\_line(' '); end loop; end;



PROGRAM 15

Write a PL/SQL program to display the employee IDs, names, and job history end dates of all employees.

Begin

for i in (select employee\_id,employee\_name,end\_date from job\_history) loop dbms\_output.put\_line('employee id: ' || i.employee\_id); dbms\_output.put\_line('name: ' || i.employee\_name);

dbms\_output.put\_line('end date: ' ||to\_char(i.end\_date, 'dd-mon-yyyy')); dbms\_output.put\_line(' '); end loop; end;

