

NAME: Akshaya Sri S

REG NO: 230701024

Ex.No.: 11		WORK WITH JOINTS
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

```
Salary after incentive : 6300
```

```
Statement processed.
```

```
0.01 seconds
```

**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;
```

```
ORA-06550: line 7, column 23:
PLS-00201: identifier 'QUOTED_VARIABLE' must be declared
ORA-06550: line 7, column 1:
PL/SQL: Statement ignored
```

### PROGRAM 3

Write a PL/SQL block to

```
Hi
Hello

Statement processed.
```

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 updatation: '||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 updatation: '||new_salary); end;
        block to
```

Before updatation: 8000

After updatation: 9000

Statement processed.

0.00 seconds

#### 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;
```

```
TRUE
FALSE
NULL VALUES in arguments
```

```
Statement processed.
```

```
0.00 seconds
```

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_e%' 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;

```

block to

```

Name starts with "D"
Name contains "Dan" followed by one character
Name contains "Daniel_Andrea"

```

Statement processed.

## 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;

```

```

Value in a : 10
Value in b : 5
Smaller number is 5
Larger number is 10

```

```
Statement processed.
```

```
0.00 seconds
```

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

PL/SQL

```

Before incentive calculation: 21000
Record(s) updated
After incentive calculation: 23500

```

```
Statement processed.
```

```

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;

```

Record(s) updated

Statement processed.

#### 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;

```

PROGRAM 10

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.



```

= 10;

declare emp_count number;
vacancy
number := 20;
begin
Select      count(*)      into  emp_count  from  employees  where
            department_id 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;
```

```

Total seats : 10
Number of employees in Department : 2
Available vacancies : 8
```

```

Statement processed.
Total seats : 20
Number of employees in Department 50 : 3
Available vacancies : 17
```

```

Statement processed.
```

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.

PROGRAM 11

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.

## PROGRAM 12

```

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;

```

---

```

employee id: 101
name: John
job title: IT_PROG
hire date: 01-jan-1994
salary: 6020
-----

```

```

employee id: 176
name: Jane
job title: HR_REP
hire date: 20-feb-2019
salary: 12500
-----

```

```

employee id: 103
name: Mike
job title: SA_MAN
hire date: 01-mar-1998
salary: 7200
-----

```

```

employee id: 104
name: Emily
job title: AC_ACCOUNT
hire date: 01-jan-1998
salary: 15000
-----

```

```

employee id: 105
name: Robert
job title: ST_CLERK
hire date: 25-jul-2018
salary: 6200
-----

```

## PROGRAM 13

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;
```

```
employee id: 101
name: John
department name: IT_PROG
-----
employee id: 176
name: Jane
department name: HR_REP
-----
employee id: 103
name: Mike
department name: SA_MAN
-----
employee id: 104
name: Emily
department name: AC_ACCOUNT
-----
employee id: 105
name: Robert
department name: ST_CLERK
-----
```

to

to

## 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;
```

```

to
job id: 101
job title: Software Engineer
minimum salary: 60000
-----
job id: 102
job title: Data Analyst
minimum salary: 50000
-----
job id: 103
job title: Project Manager
minimum salary: 70000
-----
job id: 104
job title: HR Manager
minimum salary: 55000
-----
job id: 105
job title: Marketing Specialist
minimum salary: 45000
-----

```

#### 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;

```

to

```

employee id: 201
name: James
start date: 01-jan-2010
-----

```

```

employee id: 202
name: King
start date: 01-jan-2012
-----

```

```

employee id: 203
name: Smith
start date: 01-jan-2013
-----

```

```

employee id: 204
name: Steve
start date: 01-jan-2014
-----

```

```

employee id: 205
name: Robert
start date: 01-jan-2015
-----

```

#### 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;

```



---

employee id: 201  
name: James  
end date: 10-oct-2015

-----  
employee id: 202  
name: King  
end date: 15-sep-2016

-----  
employee id: 203  
name: Smith  
end date: 20-mar-2017

-----  
employee id: 204  
name: Steve  
end date: 05-apr-2018

-----  
employee id: 205  
name: Robert  
end date: 12-may-2019

-----