PROGRAM 1

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

```
DECLARE
pl-emp-id employee_id%.TYPE=110;
pl-Salary employees.salary %TYPE;
pl-incentive NUMBER;
 BEGIN
Select salary INTO pl-salary
FROM employees
Where employee-id =pl-emp-id;
pl-incentive :=pl-salary *0.10
UPDATE employees
SET incentive =pl-incentive
 WHERE employee-id=pl-emp-id;
DBMS-OUTPUT.PUT_LINE ('Incentive for employee ID' || pl-emp-id ||
        'is'|| pl-incentive);

COMMIT;
END;
```

PROGRAM 2

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

```
DECLARE
 employee Name Varchar2 (100);
 "EmployeeID" NUMBER;
BEGIN
  employeeName := 'John Doe';
 "EmployeeID":= 40;
  DBMS-OUTPUT.PUT_LINE ('Employee Name :' || employee Name);
  DBms-OUTPUT.PUT-LINE ('Employee ID :'|| EmployeeID");

END;
```

PROGRAM 3

Write a PL/SQL block to adjust the salary of the employee whose ID 122.
Sample table: employees

```
v_employee_id NUMBER := 122;
v_salary NUMBER
v_new_salary NUMBER;
v_increase_percentage NUMBER := 0.40;

SELECT salary INTO v_salary
FROM employees
Where employee_id = v_employee_id;
v_new_salary := v_salary + (v_salary * v_increase_percentage/100);
UPDATE employees
SET salary = v_new_salary
WHERE employee_id = v_employee_id;
DBMS_OUTPUT.PUT_LINE ('Employee ID'' ||v_employee_id||'new salary:'||v_new_salary);

END;
```

PROGRAM 4

Write a PL/SQL block to 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 check_NULL
is  value1 number := 10;
    value2 number := NULL;
begin
  if value1 is not null and value 2 is null then
  dbms_output.put_line ("Both value are not NULL");
  else
      dbms_output.put_line ("Null value found');
      end if;
  end;
 BEGIN
     check_null;
END;
```

PROGRAM 5

Write a PL/SQL block to describe the usage of LIKE operator including wildcard characters and escape character.

```
declare
v_employeename employees.first_name%type;
v_employeeid NUMBER :=122;
begin
Select first_name into v_employeename
from employees
where first_name like '%e%' and employee_id = v_employeeid;
DBMS_OUTPUT.PUT_LINE (v_employeename);
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
ab number :=10;
cd number :=20;
num_small number;
num_large number;
begin
if ab > cd then
  num_small :=cd;
  num_large:=ab;

else
  num_small :=ab;
  num_large :=cd;
end if
dbms_output.put_line ('small number ='|| num_small);
dbms_output.put_line (' large number ='|| num_large);

End;
```

PROGRAM 7
Write a PL/SQL procedure to calculate the incentive on a target achieved and display the message either the record updated or not.

```
Create or Replace procedure calculate_incentive (p-emp-id
employees.employee_id %type, p-target number)
is
v-incentitive number(7,2);
v- salary employees.salary %type ;
begin
  select salary into v-salary
  from employees
  where employee-id = p-emp-id;
if p-target >= 100000 then
  v-incentive := v-salary * 0.1;
  dbms_output.put line ('Incentive of ||v-incentive||'calculated for employee ID'||
                         p-emp-id);
  ~~else~~ dbms_output.put_line ("No incentive for employee ID" ||p-emp-id);
  end if ; End;
```

PROGRAM 8
Write a PL/SQL procedure to calculate incentive achieved according to the specific sale limit.

```
Create or Replace prodecuse incentive-sale(p-emp-id employees.employee-id%
 p-sales number)
is
v- Incentive number (7,2);
begin
  if p-sales > 100000 then
   v- Incentive:= p-sales * 0.1;
UPDATE employees
SET incentive = pl-sales *0.05;
else
  v-incentive = 0;
end if
 dbms_output.put-line ('Incentive for employees ID' ||p-emp-id ||'is :'|| v-incentive);
End;

 begin incentive_sales (122, 500000);

end;
```

PROGRAM 9

Write a PL/SQL program to 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
no-of-emp number;
vacancies number := 45;
begin
select count(*) into no-of-emp from employees where department_id= 50;
if no-of-emp < vacancies then
dbms-output.put_line ('vacanciesare avaliable');
else
 dbms-output.put_line('vacantes are not available');
 END;
 /
```

PROGRAM 10

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.

```
SEt server output ON;
declare
  emp_Count Number;
   Vanancies number : = 45;
  begin
  select count (*) Into emp - count from employee where
                      department = 50;
  DBMS - output .PUt - LINE ("employees Deparo" ||
                       emp - count || Vacancies ||
 (Vancandies - emp_count));
  END;
 /
```

PROGRAM 11

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

```
Set Server output ON;
 begin
 for rec In(select employee_id, name, Job-title, hire date,
                                            Salary
 FROM DBMS _OUTPUT (' ID: "|| rec.employee-id ||
            ';Name:'|| rec-name || ', Job title :'||rec.job_title ||
            'Hire Date :"|| rec.hire -date ||
            'Salary :'||rec.salary);
 END LOOP;
 END;
```

PROGRAM 12
Write a PL/SQL program to display the employee IDs, names, and department names of all employees.

```
SET SERVER OUTPUT ON;
BEGIN
FOR REC IN(select e.employee-id, e.name,
     d.department_name
          FROM employees e
        .. JOIN department d on e.department-id=d.department
 DBMS-OUTPUT_PUT-LINE ('ID:'|| rec.employees_id ||
                  ', NAME:'||rec.name ||
                  ', Department:'||rec.department-name);
   END LOOP;
   END;
```

100

PROGRAM 13
Write a PL/SQL program to display the job IDs, titles, and minimum salaries of all jobs.

```
SET SERVER OUTPUT. ON ;
 BEGIN
FOR recIN (Select job_id, job-title, min_salary
                    FROM) LOOP
DBMS-OUTPUT. PUT.LINE ('JobID: ||rec.job-id||
                    ', Title : '||re .job - tittel|
                    ', min salary :' || rec .min - salary);

    End loop;
  End;
  /
```

PROGRAM 14
Write a PL/SQL program to display the employee IDs, names, and job history start dates of all employees.

```
Set Serveroutput ON:
  Begin
       for rec (select e . employee - id - j - employ-
                                                      id)
DBMS- Output. Put - Line ( 'ID: ' ||rec. employe
                          ',Name :' || rec . name ||
                        ', Job Start Date : || rec . start_date);

ENDloop;
END;
```

PROGRAM 15

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

```
SET SERVER OUT PUT ON ;
 BEGIN
   FOR REC IN (SELECT e.employee_id, e_name, j_end_da
     FROM employee e
    JOIN job_historg jon e.employee_id = j.employee.id)
DBMS_OUTPUT . PUT_LINE ('ID'||rec.employee_id||
                              ', Name :'||rec.name||
                               ', Job End date :'||rec.end.date);

END LOOP ;
 END
```

Evaluation Procedure	Marks awarded
PL/SQL Procedure(5)	
Program/Execution (5)	
Viva(5)	
Total (15)	
Faculty Signature	