

PROGRAM 1

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

Declare

```
v-emp-id NUMBER := 110;  
v-incentive-amount NUMBER;  
v-salary NUMBER;
```

Begin

```
select salary into v-salary  
from employees  
where employee-id = v-employee-id  
v-incentive-amount := v-salary * 0.10;  
DBMS-Output-put_line ('Incentive calculated  
for employee' || v-employee  
|| ': ' || v-incentive-amount);
```

END;

PROGRAM 2

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

```
SET server output ON;

DECLARE
    "Name" varchar2(20) = 'virat';
BEGIN
    DBMS_OUTPUT.put_line(name);
END;
```

PROGRAM 3

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

Sample table: employees

```
SET SERVER OUTPUT ON;

DECLARE
    v_old_salary employees.salary%TYPE;
BEGIN
    SELECT salary INTO v_old_salary
    FROM employees
    WHERE employee_id = 122;


    UPDATE EMPLOYEES
    SET salary = v_old_salary * 1.1;

    DBMS_OUTPUT.PUT_LINE ('old salary: ' || v_old_salary);
    DBMS_OUTPUT.PUT_LINE ('new salary: ' || (v_old_salary * 1.1));

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE ('Employee ID 122 not found');

END;
```

3



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_and_and (
```

```
    p-num1 IN number,
```

```
    p-num2 IN number
```

```
)
```

```
is
```

```
    v-is-num1-null BOOLEAN;
```

```
    v-is-num2-null BOOLEAN;
```

```
BEGIN
```

```
    v-is-num1-null = p-num1 is NULL;
```

```
    v-is-num2-null = p-num2 is NULL;
```

```
    v-and-result; = (p-num1 is not null) and (p-num2 is not null)
```

```
    DBMS_output.put_line ('Number 1 is NULL: ' ||
```

```
    DBMS_output_line ('Result of (Number 1 is not null and  
    Number 2 is not null);')
```

```
END
```

```
/
```

PROGRAM 5

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

DECLARE

v_name varchar2(100);

Begin

DBMS_output.put_line

FOR rec IN (SELECT name FROM employees where name
like '%_%' escape '\') LOOP

DBMS_output.put_line(rec.name);

END LOOP;

DBMS_output.put_line('-- using Escape (names containing
a literal :%) - -');

FOR rec IN (select name from employees where name
and like '%_%' escape '\') LOOP

DBMS_output.put_line(rec.name);

END LOOP;

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

```
num1 NUMBER := 10;  
num2 NUMBER := 20;  
num_small NUMBER;  
num_large NUMBER;
```

BEGIN

```
IF num1 < num2 THEN
```

```
    num_small := num1;
```

```
    num_large := num2;
```

```
END IF;
```


```
DBMS_OUTPUT.put_line ('Small number:' || num_small);
```

```
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 TABLE employees (  
    employee_id NUMBER Primary Key,  
    salary number,  
    target_achieved number  
);  
  
INSERT INTO employees values (101, 50000, 1);  
INSERT INTO employees values (102, 60000, 0);  
  
COMMIT;
```



PROGRAM 8

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

```
CREATE OR REPLACE PROCEDURE calculate_incentive (  
    p_sales_amount IN NUMBER  
)  
IS  
    v_incentive NUMBER;  
    v_sales_limit CONSTANT NUMBER := 1000;  
    v_incentive_rate CONSTANT NUMBER := 0.10;  
  
BEGIN  
    IF p_sales_amount > v_sales_limit THEN  
  
    ELSE  
        v_incentive := 0;  
    END IF;  
  
EXCEPTION  
    WHEN OTHERS THEN  
        DBMS_OUTPUT.PUT_LINE ('An error occurred')  
  
END calculate_incentive;  
  
/
```

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
    emp_count NUMBER;
    total_vacancies CONSTANT number := 45;
BEGIN
    SELECT count (*)
    into v_employee_count
    from employees
    where dept_id = 50;
    DBMS_output.put_line ('Total employees in department 50: ' || v_employee_count);
    IF v_employee_count < total_vacancies THEN
        DBMS_output.put_line ('Department 50 has vacancies');
    ELSE
        DBMS_output.put_line ('Department 50 has no vacancies');
    END IF;
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.

declare

v_dept_id number; = 10;

v_emp_count number;

v_total_positions number := 5;

v_vacancies number;

begin

select count (*)

into v_emp_count
from employees

where department_id = v_dept_id;

v_vacancies = v_total_positions

if v_vacancies > 0 then

dbms_output.put_line('Department ' || v_dept_id);

end if;

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;
declare
    v_employee_id
    v_first_name
    v_last_name
    v_job_title
    v_hire_date
    v_salary

    cursor emp_cursor is
        select employee_id, first_name, last_name, job_title,
               hire_date, salary
        from employees
begin
    open emp_cursor
    loop
        fetch emp_cursor into v_employee_id,
                               v_first_name,
                               v_last_name,
                               v_job_title,
                               v_hire_date,
                               v_salary;
        dbms_output.put_line ('ID : ' || v_employee_id);
    end loop;
    close emp_cursor;
end;
```

PROGRAM 12

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

```
Set server output on;

declare
    dbms_output.put_line('Employee ID/Name/Department
                           Name');
    dbms_output.put_line('---');

open c_employee_info;

loop
    fetch c_employee_info into v_employee_id,
                               v_f_name, v_l_name, v_dept_name
    exit when c_employee_info not found;

    dbms_output.put_line(v_employee_id || ' ' || v_f_name
                          || ' ' || v_l_name || ' ' ||
                          v_dept_name);

end loop;
```

PROGRAM 13

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

```
set server output on;

declare
    v_job_title
    v_job_id
    v_min_salary jobs

begin
    for job_rec in (select 'job-id', job_title, min
                        salary)
    loop
        v_job_id = job_rec.job_id;
        v_job_title = job_rec.job_title;
        v_min_salary := job_rec.min_salary;

        dbms_output.put_line ('Job-ID : ' || job_id || Title
                                || v_job_title || ' minsalary : '
                                || v_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.

Begin

open c - employee - history

loop

fetch c - employee - history into v - employee - id ,

v - f - name , v - l - name , v - st - date ;

exit when c - employee - history % not found ;

dbms - output - put line ('Employee ID : ' || v - employee - id

, name : ' || v - f - name || ' ' || v - l - name

' , start date : ' || v - st - date) ;

end loop ;

close c - employee - history ;

exception

when others then

dbms - output - put line ('An error occurred ') ,

END ;

/

PROGRAM 15

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

Begin


dbms_output.put_line ('Employee - ID / Employee name

dbms_output.put_line ('---');

open c_employee_history;

loop

fetch c_employee_history into v_employee_id,

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

v_full_name,
v_end_date;

exit when c_employee_history %notfound;

dbms_output.put_line (v_employee_id || ' ' ||

v_full_name || ' ' || v_end_date);

end loop;

END;