

PROGRAM I

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 * 110;
    DBMS_OUTPUT.PUT_LINE ("old_salary : " || v_old_salary);
    DBMS_OUTPUT.PUT_LINE ('new salary : ' || (v_old_salary * 110));
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE ('Employee ID 122 not found');
END;
```

y

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
```

```
),
```

```
    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('Number1 is NULL: ' ||)
```

```
    DBMS_OUTPUT.PUT_LINE('Result of (Number1 is not null and  
    Number2 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 '%_l_%' 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 '%.l%.' 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
        v_incentive := v_incentive_rate * p_sales_amount;
    ELSE
        v_incentive := 0;
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE ('An error occurred');
END calculate_incentive;
```

PROGRAM 9

Write a PLISQL 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 (e)

INTO v_employee_count

FROM employees

WHERE dept_id = 50;

DEMS_OUTPUT.PUT_LINE ('Total employees in department 50');

IF v_employee_count < vacancies THEN

DEMS_OUTPUT.PUT_LINE ('Department 50 has vacancies');

ELSE

DEMS_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 - v_emp_count;
    if v_vacancies > 0 then
        dbms_output_put_line ('Department '||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,
                        dbms_output.put_line ('ID : ' || v_employee_id);
    end loop;
    close 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');
    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
        "|| 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
                     from jobs)
    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 : ' || v_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,  
            v_full_name,  
            v_end_date;
```

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

exit when c_employee_history > not found;

```
    dbms_output.put_line ('v_employee_id ||' ||  
        v_full_name ||' ||' || v_end_date);  
end loop;  
END;
```