Ex.No.: 11		**PL SQL PROGRAMS**
Date:		

Here are the PL/SQL programs as requested:

PROGRAM 1
PL/SQL block to calculate the incentive of an employee whose ID is 110.
```sql
DECLARE
   v_employee_id NUMBER := 110;
   v_salary employees.salary%TYPE;
   v_incentive NUMBER;
BEGIN
   SELECT salary INTO v_salary FROM employees WHERE employee_id = v_employee_id;
   v_incentive := v_salary * 0.10;  -- Assuming 10% incentive rate
   DBMS_OUTPUT.PUT_LINE('Incentive for employee ID ' || v_employee_id || ' is: ' ||
v_incentive);
END;
/
```

PROGRAM 2
PL/SQL block to show an invalid case-insensitive reference to a quoted and unquoted user-defined identifier.
```sql
DECLARE
   "MyVariable" NUMBER := 100;  -- Quoted identifier
   myvariable NUMBER := 200;    -- Unquoted identifier
BEGIN
   DBMS_OUTPUT.PUT_LINE('Value of "MyVariable" is ' || "MyVariable");
   DBMS_OUTPUT.PUT_LINE('Value of myvariable is ' || myvariable);
   -- Uncommenting the next line will cause an error, as quoted and unquoted identifiers are
treated differently.
```

```
   -- DBMS_OUTPUT.PUT_LINE('This will cause an error: ' || MyVariable);
END;
/
```

PROGRAM 3
PL/SQL block to adjust the salary of the employee whose ID is 122.
```sql
DECLARE
   v_employee_id NUMBER := 122;
   v_new_salary employees.salary%TYPE := 5500;  -- New salary value
BEGIN
   UPDATE employees
   SET salary = v_new_salary
   WHERE employee_id = v_employee_id;

   IF SQL%ROWCOUNT > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Salary updated for employee ID ' || v_employee_id);
   ELSE
      DBMS_OUTPUT.PUT_LINE('No employee found with ID ' || v_employee_id);
   END IF;
END;
/
```

PROGRAM 4
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`.
```sql
DECLARE
   v_value1 NUMBER := 10;
   v_value2 NUMBER := NULL;
BEGIN
   IF v_value1 IS NOT NULL AND v_value2 IS NOT NULL THEN
      DBMS_OUTPUT.PUT_LINE('Both values are NOT NULL');
   ELSE
      DBMS_OUTPUT.PUT_LINE('One or both values are NULL');
   END IF;
END;
/
```

~~~~
```

---

### PROGRAM 5
**PL/SQL block to describe the usage of the `LIKE` operator, including wildcard characters and
escape character.**
```sql
DECLARE
   v_name employees.last_name%TYPE := 'Smi%';
BEGIN
   FOR rec IN (SELECT last_name FROM employees WHERE last_name LIKE v_name
ESCAPE '\') LOOP
      DBMS_OUTPUT.PUT_LINE('Employee Name: ' || rec.last_name);
   END LOOP;
END;
/
```

---

### PROGRAM 6
**PL/SQL program to arrange the numbers of two variables so that the smaller number is stored
in `num_small` and the larger in `num_large`.**
```sql
DECLARE
   num1 NUMBER := 45;
   num2 NUMBER := 30;
   num_small NUMBER;
   num_large NUMBER;
BEGIN
   IF num1 < num2 THEN
      num_small := num1;
      num_large := num2;
   ELSE
      num_small := num2;
      num_large := num1;
   END IF;

   DBMS_OUTPUT.PUT_LINE('Small Number: ' || num_small);
   DBMS_OUTPUT.PUT_LINE('Large Number: ' || num_large);
END;
/
```
~~~~

PROGRAM 7
PL/SQL procedure to calculate the incentive on a target achieved and display a message indicating whether the record was updated.
```sql
CREATE OR REPLACE PROCEDURE calculate_incentive(p_employee_id NUMBER, p_target
NUMBER) AS
   v_incentive NUMBER;
BEGIN
   IF p_target >= 1000 THEN
      v_incentive := p_target * 0.05;  -- Example incentive calculation
      DBMS_OUTPUT.PUT_LINE('Incentive calculated: ' || v_incentive);
      DBMS_OUTPUT.PUT_LINE('Record updated.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Target not achieved. No update.');
   END IF;
END calculate_incentive;
/
```

PROGRAM 8
PL/SQL procedure to calculate incentive achieved according to the specific sale limit.
```sql
CREATE OR REPLACE PROCEDURE calculate_sales_incentive(p_sales_amount NUMBER)
AS
   v_incentive NUMBER;
BEGIN
   IF p_sales_amount >= 5000 THEN
      v_incentive := p_sales_amount * 0.10;
   ELSIF p_sales_amount >= 3000 THEN
      v_incentive := p_sales_amount * 0.07;
   ELSE
      v_incentive := p_sales_amount * 0.05;
   END IF;

   DBMS_OUTPUT.PUT_LINE('Incentive achieved: ' || v_incentive);
END calculate_sales_incentive;
/
```

Name: Ram Haygrev S Roll no: 231901039

These PL/SQL blocks cover a range of tasks from variable handling, control structures, and conditional checks, to defining and using procedures with parameters.