PL/SQL Practical Scripts - Complete Set

---

1. PL/SQL Block using Nested IF to Grade a Student

```
DECLARE
  v_marks  NUMBER := 78;
  v_grade  CHAR(1);
BEGIN
  IF v_marks >= 90 THEN
    v_grade := 'A';
  ELSE
    IF v_marks >= 75 THEN
      v_grade := 'B';
    ELSE
      IF v_marks >= 60 THEN
        v_grade := 'C';
      ELSE
        v_grade := 'F';
      END IF;
    END IF;
  END IF;
  DBMS_OUTPUT.PUT_LINE('Marks = '||v_marks||', Grade = '||v_grade);
END;
/
```

---

2. PL/SQL Program to Find Factorial of a Number

```
DECLARE
  n         PLS_INTEGER := 6;
  fact_value NUMBER := 1;
```

```
BEGIN
  IF n < 0 THEN
        DBMS_OUTPUT.PUT_LINE('Factorial not defined for negative numbers');
  ELSE
    FOR i IN 1..n LOOP
      fact_value := fact_value * i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Factorial('||n||') = '||fact_value);
  END IF;
END;
/
```

---

3. Create Table and Insert Records

```
CREATE TABLE employee (
  emp_id     NUMBER PRIMARY KEY,
  emp_name   VARCHAR2(50) NOT NULL,
  department VARCHAR2(30),
  salary     NUMBER(10,2)
);

INSERT INTO employee VALUES (101,'Ravi Sharma','HR',25000.00);
INSERT INTO employee VALUES (102,'Asha Mehta','IT',45000.00);
INSERT INTO employee VALUES (103,'John Dsouza','Finance',15000.00);
INSERT INTO employee VALUES (104,'Neha Patel','IT',55000.00);
INSERT INTO employee VALUES (105,'Karan Joshi','Marketing',9500.00);
COMMIT;
```

---

4. Bonus Calculation (10%) Display Name + Bonus

```
BEGIN
  FOR r IN (SELECT emp_name, salary FROM employee) LOOP
        DBMS_OUTPUT.PUT_LINE(r.emp_name || ' -> Bonus(10%) = ' ||
TO_CHAR(r.salary * 0.10,'999,990.99'));
  END LOOP;
END;
/

-----------------------------------------
5. Cursor - Employees with Salary > 20000
-----------------------------------------
DECLARE
  CURSOR c_highpay IS
     SELECT emp_id, emp_name, salary FROM employee WHERE salary >
20000 ORDER BY salary DESC;

  v_row c_highpay%ROWTYPE;
BEGIN
  OPEN c_highpay;
  LOOP
    FETCH c_highpay INTO v_row;
    EXIT WHEN c_highpay%NOTFOUND;
     DBMS_OUTPUT.PUT_LINE(v_row.emp_id || ' - ' || v_row.emp_name || ' :
' || v_row.salary);
  END LOOP;
  CLOSE c_highpay;
END;
/

-----------------------------------------
6. Department-wise Employee Count
-----------------------------------------
BEGIN
```

```
  FOR r IN (SELECT NVL(department,'(No Dept)') AS dept, COUNT(*) AS total FROM employee GROUP BY department ORDER BY dept) LOOP
    DBMS_OUTPUT.PUT_LINE(r.dept || ' -> ' || r.total || ' employee(s)');
  END LOOP;
END;
/

-----------------------------------------
7. Cursor - Employee Names Starting with 'N'
-----------------------------------------
DECLARE
     CURSOR c_names_starting_with_n IS SELECT emp_name FROM employee WHERE UPPER(emp_name) LIKE 'N%';
  v_name employee.emp_name%TYPE;
BEGIN
  OPEN c_names_starting_with_n;
  LOOP
    FETCH c_names_starting_with_n INTO v_name;
    EXIT WHEN c_names_starting_with_n%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE(v_name);
  END LOOP;
  CLOSE c_names_starting_with_n;
END;
/

-----------------------------------------
8. BEFORE Trigger - Block Low Salaries
-----------------------------------------
CREATE OR REPLACE TRIGGER trg_block_low_salary
BEFORE INSERT OR UPDATE OF salary ON employee
FOR EACH ROW
BEGIN
  IF :NEW.salary < 10000 THEN
```

```
        RAISE_APPLICATION_ERROR(-20001, 'Salary below minimum threshold (10,000) is not allowed.');
  END IF;
END;
/

-----------------------------------------
9. FUNCTION - Annual Salary by Employee ID
-----------------------------------------
CREATE OR REPLACE FUNCTION get_annual_salary(p_emp_id IN employee.emp_id%TYPE)
RETURN NUMBER
IS
  v_monthly employee.salary%TYPE;
BEGIN
   SELECT salary INTO v_monthly FROM employee WHERE emp_id = p_emp_id;
  RETURN v_monthly * 12;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
   RETURN NULL;
END;
/

-----------------------------------------
10. PROCEDURE - Update Salary by Percentage
-----------------------------------------
CREATE OR REPLACE PROCEDURE increase_salary_by_percent(
  p_emp_id  IN employee.emp_id%TYPE,
  p_percent IN NUMBER
)
AS
BEGIN
```

```
  UPDATE employee
  SET salary = salary * (1 + (p_percent/100))
  WHERE emp_id = p_emp_id;

  IF SQL%ROWCOUNT = 0 THEN
    RAISE_APPLICATION_ERROR(-20002, 'Employee not found.');
  END IF;
END;
/
```

---

11. PACKAGE - Employee Operations (Function + Procedure)

```
CREATE OR REPLACE PACKAGE emp_ops AS
    FUNCTION  get_annual_salary(p_emp_id  IN  employee.emp_id%TYPE)
RETURN NUMBER;
                PROCEDURE        increase_salary_by_percent(p_emp_id        IN
employee.emp_id%TYPE, p_percent IN NUMBER);
END emp_ops;
/

CREATE OR REPLACE PACKAGE BODY emp_ops AS
  FUNCTION get_annual_salary(p_emp_id IN employee.emp_id%TYPE)
  RETURN NUMBER IS
    v_sal employee.salary%TYPE;
  BEGIN
        SELECT  salary  INTO  v_sal  FROM  employee  WHERE  emp_id  =
p_emp_id;
    RETURN v_sal * 12;
  EXCEPTION WHEN NO_DATA_FOUND THEN
    RETURN NULL;
  END get_annual_salary;
```

```
    PROCEDURE increase_salary_by_percent(p_emp_id IN
employee.emp_id%TYPE, p_percent IN NUMBER)
  IS
  BEGIN
    UPDATE employee
    SET salary = salary * (1 + (p_percent/100))
    WHERE emp_id = p_emp_id;

    IF SQL%ROWCOUNT = 0 THEN
      RAISE_APPLICATION_ERROR(-20002, 'Employee not found.');
    END IF;
  END increase_salary_by_percent;
END emp_ops;
/
```