# SUSHMIT PARTAKKE

# 23070521156

# SEM 4

# PRAC 6 FINAL

## Tasks on PL/SQL Basics with Database

**Task 1:** Write a PL/SQL block to insert a new employee into the `employees` table.

**Table:** `employees(emp_id, emp_name, salary, department)`

Insert an employee with `emp_id = 101`, `emp_name = 'John Doe'`, `salary = 5000`, `department = 'IT'`.

**Task 2:** Create a PL/SQL block to retrieve and display all employee names from the `employees` table.

**Task 3:** Write a PL/SQL block to update the salary of an employee whose `emp_id = 101` by increasing it by **10%**.

**Task 4:** Create a PL/SQL block to delete an employee whose `emp_id = 105`.

**Task 5:** Display the count of employees in the `employees` table.

## Tasks on Conditional Statements with Database

**Task 6:** Write a PL/SQL block that checks if an employee's salary is above **5000**. If yes, print `"High Salary"`; otherwise, print `"Low Salary"`.

**Task 7:** Fetch the department of an employee based on `emp_id` and print:

- `"IT Department"` if in **IT**,
- `"HR Department"` if in **HR**,

- `"Other Department"` otherwise.

**Task 8:** Use a `CASE` statement to categorize employees based on salary:

- **Above 8000 → "Senior Level"**
- **5000-8000 → "Mid Level"**
- **Below 5000 → "Junior Level"**

**Task 9:** If an employee's department is **Sales**, increase their salary by **5%**.

**Task 10:** Check if an employee with `emp_id = 110` exists. If not, insert a new record.

## Tasks on Loops with Database

**Task 11:** Use a **FOR LOOP** to print all employees' names from the `employees` table.

**Task 12:** Write a **LOOP** to insert **5 new employees** into the `employees` table.

**Task 13:** Use a **WHILE LOOP** to increase the salary of all employees earning less than **4000** by **20%**.

**Task 14:** Create a **FOR LOOP** that prints the first **3 departments** from the `departments` table.

**Task 15:** Write a **LOOP** to delete employees who have not updated their records in the last **5 years** (assuming there's a `last_updated` column).

**Task 16:** Use a **LOOP** to find the employee with the highest salary in the `employees` table.

**Task 17:** Fetch and display all employees in a specific department using a **WHILE LOOP**.

**Task 18:** Write a **LOOP** to insert **10 new customers** into a `customers` table.

**Task 19:** Use a **FOR LOOP** to display the top **5 highest-paid employees** from the `employees` table.

**Task 20:** Write a **LOOP** to find and delete duplicate employee records in the `employees` table.

```
CREATE TABLE employees (
    emp_id NUMBER PRIMARY KEY,
    emp_name VARCHAR2(100),
    salary NUMBER,
    department VARCHAR2(50),
    last_updated DATE DEFAULT SYSDATE
);


INSERT INTO employees (emp_id, emp_name, salary, department) VALUES (101, 'Alice', 5000,
'HR');
INSERT INTO employees (emp_id, emp_name, salary, department) VALUES (102, 'Bob', 4500,
'IT');
INSERT INTO employees (emp_id, emp_name, salary, department) VALUES (103, 'Charlie',
6000, 'Finance');
INSERT INTO employees (emp_id, emp_name, salary, department) VALUES (104, 'David',
5200, 'Sales');
INSERT INTO employees (emp_id, emp_name, salary, department) VALUES (105, 'Emma',
4800, 'Marketing');

CREATE TABLE departments (
    dept_id NUMBER PRIMARY KEY,
    department VARCHAR2(50)
);


INSERT INTO departments (dept_id, department) VALUES (1, 'HR');
INSERT INTO departments (dept_id, department) VALUES (2, 'IT');
INSERT INTO departments (dept_id, department) VALUES (3, 'Finance');
INSERT INTO departments (dept_id, department) VALUES (4, 'Sales');
INSERT INTO departments (dept_id, department) VALUES (5, 'Marketing');

CREATE TABLE customers (
    cust_id NUMBER PRIMARY KEY,
    cust_name VARCHAR2(100)
);


INSERT INTO customers (cust_id, cust_name) VALUES (1, 'Customer A');
INSERT INTO customers (cust_id, cust_name) VALUES (2, 'Customer B');
INSERT INTO customers (cust_id, cust_name) VALUES (3, 'Customer C');
INSERT INTO customers (cust_id, cust_name) VALUES (4, 'Customer D');
INSERT INTO customers (cust_id, cust_name) VALUES (5, 'Customer E');
```

```
COMMIT;

-- Task 1
INSERT INTO employees (emp_id, emp_name, salary, department) VALUES (106, 'Frank',
5500, 'HR');

-- Task 2
DECLARE
   CURSOR emp_cursor IS SELECT emp_name FROM employees;
   emp_name_var employees.emp_name%TYPE;
BEGIN
   OPEN emp_cursor;
   LOOP
      FETCH emp_cursor INTO emp_name_var;
      EXIT WHEN emp_cursor%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE(emp_name_var);
   END LOOP;
   CLOSE emp_cursor;
END;
/

-- Task 3
UPDATE employees SET salary = salary * 1.10 WHERE emp_id = 101;

-- Task 4
DELETE FROM employees WHERE emp_id = 105;

-- Task 5
DECLARE
   emp_count NUMBER;
BEGIN
   SELECT COUNT(*) INTO emp_count FROM employees;
   DBMS_OUTPUT.PUT_LINE('Total Employees: ' || emp_count);
END;
/

-- Task 6
DECLARE
   emp_salary NUMBER;
BEGIN
   SELECT salary INTO emp_salary FROM employees WHERE emp_id = 101;
   DBMS_OUTPUT.PUT_LINE('Salary: ' || emp_salary);
END;
/
```

```
-- Task 7
DECLARE
   emp_dept VARCHAR2(50);
BEGIN
   SELECT department INTO emp_dept FROM employees WHERE emp_id = 101;
   DBMS_OUTPUT.PUT_LINE('Department: ' || emp_dept);
END;
/

-- Task 8
UPDATE employees SET salary = salary * 1.05 WHERE department = 'Sales';

-- Task 9
DECLARE
   emp_exists NUMBER;
BEGIN
   SELECT COUNT(*) INTO emp_exists FROM employees WHERE emp_id = 110;
   IF emp_exists > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Employee exists.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('Employee does not exist.');
   END IF;
END;
/

-- Task 10
BEGIN
   FOR emp_rec IN (SELECT emp_name FROM employees) LOOP
      DBMS_OUTPUT.PUT_LINE(emp_rec.emp_name);
   END LOOP;
END;
/

-- Task 11
DECLARE
   i NUMBER := 6;
BEGIN
   WHILE i <= 10 LOOP
      INSERT INTO customers (cust_id, cust_name) VALUES (i, 'Customer ' || i);
      i := i + 1;
   END LOOP;
   COMMIT;
END;
/
```

```
-- Task 12
BEGIN
   FOR emp_rec IN (SELECT emp_name, salary FROM employees ORDER BY salary DESC
FETCH FIRST 5 ROWS ONLY) LOOP
      DBMS_OUTPUT.PUT_LINE(emp_rec.emp_name || ': ' || emp_rec.salary);
   END LOOP;
END;
/

-- Task 13
DELETE FROM employees WHERE ROWID NOT IN (SELECT MIN(ROWID) FROM
employees GROUP BY emp_id);

-- Commit changes
COMMIT;
```

OUTPUT

```
Alice
Bob
Charlie
David
Emma
Frank
Total Employees: 5
Salary: 5500
Department: HR
Employee does not exist.
Alice
Bob
Charlie
David
Frank
Charlie: 6000
Alice: 5500
Frank: 5500
David: 5460
Bob: 4500
```