Name:Sharvayu Zade
PRN:23070521135
SEC:B(B2)

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'`.

```
SQL> BEGIN
  2      INSERT INTO employees (emp_id, emp_name, salary, department)
  3      VALUES (101, 'John Doe', 5000, 'IT');
  4  END;
  5  /
    INSERT INTO employees (emp_id, emp_name, salary, department)
```

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

```
SQL> DECLARE
  2      emp_name VARCHAR2(20);
  3  BEGIN
  4      FOR emp IN (SELECT emp_name FROM employees) LOOP
  5          DBMS_OUTPUT.PUT_LINE(emp.emp_name);
  6      END LOOP;
  7  END;
  8  /
    FOR emp IN (SELECT emp_name FROM employees) LOOP
```

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

```
SQL> BEGIN
  2       UPDATE employees
  3       SET salary = salary * 1.10
  4       WHERE emp_id = 101;
  5  END;
  6  /
    UPDATE employees
```

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

```
SQL> DECLARE
  2      emp_count NUMBER;
  3  BEGIN
  4      SELECT COUNT(*) INTO emp_count
  5      FROM employees;
  6      DBMS_OUTPUT.PUT_LINE('Number of employees: ' || emp_count);
  7  END;
  8  /
    FROM employees;
```

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

```
SQL> DECLARE
  2      emp_count NUMBER;
  3  BEGIN
  4      SELECT COUNT(*) INTO emp_count
  5      FROM employees;
  6      DBMS_OUTPUT.PUT_LINE('Number of employees: ' || emp_count);
  7  END;
  8  /
    FROM employees;
```

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"`.

```
SQL> DECLARE
  2       salary NUMBER;
  3   BEGIN
  4       SELECT salary INTO salary
  5       FROM employees
  6       WHERE emp_id = 101;
  7
  8       IF salary > 5000 THEN
  9           DBMS_OUTPUT.PUT_LINE('High Salary');
 10       ELSE
 11           DBMS_OUTPUT.PUT_LINE('Low Salary');
 12       END IF;
 13   END;
 14   /
    FROM employees
```

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.

```
SQL> DECLARE
  2      department VARCHAR2(20);
  3  BEGIN
  4      SELECT department INTO department
  5      FROM employees
  6      WHERE emp_id = 101;
  7
  8      IF department = 'IT' THEN
  9          DBMS_OUTPUT.PUT_LINE('IT Department');
 10      ELSIF department = 'HR' THEN
 11          DBMS_OUTPUT.PUT_LINE('HR Department');
 12      ELSE
 13          DBMS_OUTPUT.PUT_LINE('Other Department');
 14      END IF;
 15  END;
 16  /
    FROM employees
```

Task 8: Use a `CASE` statement to categorize employees based on salary: ● **Above 8000 → "Senior Level"**

● **5000-8000 → "Mid Level"**

● **Below 5000 → "Junior Level"**

```
SQL> DECLARE
  2      salary NUMBER;
  3      category VARCHAR2(20);
  4  BEGIN
  5      SELECT salary INTO salary
  6      FROM employees
  7      WHERE emp_id = 101;
  8
  9      category := CASE
 10          WHEN salary > 8000 THEN 'Senior Level'
 11          WHEN salary BETWEEN 5000 AND 8000 THEN 'Mid Level'
 12          ELSE 'Junior Level'
 13      END;
 14
 15      DBMS_OUTPUT.PUT_LINE('Category: ' || category);
 16  END;
 17  /
    FROM employees
```

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.

```
SQL> BEGIN
  2      UPDATE employees
  3      SET salary = salary * 1.05
  4      WHERE department = 'Sales';
  5  END;
  6  /
    UPDATE employees
```

Tasks on Loops with Database

```
SQL> DECLARE
  2      emp_exists NUMBER;
  3  BEGIN
  4      SELECT COUNT(*) INTO emp_exists
  5      FROM employees
  6      WHERE emp_id = 110;
  7
  8      IF emp_exists = 0 THEN
  9          INSERT INTO employees (emp_id, emp_name, salary, department)
 10          VALUES (110, 'New Employee', 5000, 'IT');
 11      END IF;
 12  END;
 13  /
```

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

```
SQL> BEGIN
  2      FOR emp IN (SELECT emp_name FROM employees) LOOP
  3          DBMS_OUTPUT.PUT_LINE(emp.emp_name);
  4      END LOOP;
  5  END;
  6  /
    FOR emp IN (SELECT emp_name FROM employees) LOOP
                                                *
```

table.

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

```
SQL> DECLARE
  2      i NUMBER := 1;
  3  BEGIN
  4      WHILE i <= 5 LOOP
  5          INSERT INTO employees (emp_id, emp_name, salary, department)
  6          VALUES (i + 100, 'New Employee ' || i, 5000, 'IT');
  7          i := i + 1;
  8      END LOOP;
  9  END;
 10  /
```

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

```
SQL> DECLARE
  2      i NUMBER := 1;
  3      salary NUMBER;
  4  BEGIN
  5      WHILE i <= (SELECT COUNT(*) FROM employees) LOOP
  6          SELECT salary INTO salary
  7          FROM (SELECT salary, ROW_NUMBER() OVER (ORDER BY emp_id) AS row_num FROM employees)
  8          WHERE row_num = i;
  9
 10          IF salary < 4000 THEN
 11              UPDATE employees
 12              SET salary = salary * 1.20
 13              WHERE emp_id = (SELECT emp_id FROM (SELECT emp_id, ROW_NUMBER() OVER (ORDER BY emp_id) AS row_num FROM employees) WHERE row_num = i
);
 14          END IF;
 15
 16          i := i + 1;
 17      END LOOP;
 18  END;
 19  /
```

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

```
SQL> DECLARE
  2      department VARCHAR2(20);
  3  BEGIN
  4      FOR dept IN (SELECT department FROM departments FETCH FIRST 3 ROWS ONLY) LOOP
  5          DBMS_OUTPUT.PUT_LINE(dept.department);
  6      END LOOP;
  7  END;
  8  /
```

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).

```
SQL> DECLARE
  2      i NUMBER := 1;
  3      last_updated DATE;
  4  BEGIN
  5      WHILE i <= (SELECT COUNT(*) FROM employees) LOOP
  6          SELECT last_updated INTO last_updated
  7          FROM (SELECT last_updated, ROW_NUMBER() OVER (ORDER BY emp_id) AS row_num FROM employees)
  8          WHERE row_num = i;
  9
 10          IF last_updated < ADD_MONTHS(SYSDATE, -60) THEN
 11              DELETE FROM employees
 12              WHERE emp_id = (SELECT emp_id FROM (SELECT emp_id, ROW_NUMBER() OVER (ORDER BY emp_id) AS row_num FROM employees) WHERE row_num = i
);
 13          END IF;
 14
 15          i := i + 1;
 16      END LOOP;
 17  END;
 18  /
```

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

table.

```
SQL> DECLARE
  2      max_salary NUMBER := 0;
  3      emp_id NUMBER;
  4  BEGIN
  5      FOR emp IN (SELECT emp_id, salary FROM employees) LOOP
  6          IF emp.salary > max_salary THEN
  7              max_salary := emp.salary;
  8              emp_id := emp.emp_id;
  9          END IF;
 10      END LOOP;
 11
 12      DBMS_OUTPUT.PUT_LINE('Employee with highest salary: ' || emp_id);
 13  END;
 14  /
```

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

```
SQL> DECLARE
  2      i NUMBER := 1;
  3      emp_name VARCHAR2(20);
  4      department VARCHAR2(20) := 'IT';
  5  BEGIN
  6      WHILE i <= (SELECT COUNT(*) FROM employees) LOOP
  7          SELECT emp_name INTO emp_name
  8          FROM (SELECT emp_name, ROW_NUMBER() OVER (ORDER BY emp_id) AS row_num, department AS dept FROM employees)
  9          WHERE row_num = i AND dept = department;
 10
 11          DBMS_OUTPUT.PUT_LINE(emp_name);
 12
 13          i := i + 1;
 14      END LOOP;
 15  END;
 16  /
```

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

```
SQL> DECLARE
  2      i NUMBER := 1;
  3  BEGIN
  4      WHILE i <= 10 LOOP
  5          INSERT INTO customers (cust_id, name, address)
  6          VALUES (i, 'Customer ' || i, 'Address ' || i);
  7          i := i + 1;
  8      END LOOP;
  9  END;
 10  /
```

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

table.

```
SQL> DECLARE
  2      salary NUMBER;
  3  BEGIN
  4      FOR emp IN (SELECT salary FROM employees ORDER BY salary DESC FETCH FIRST 5 ROWS ONLY) LOOP
  5          DBMS_OUTPUT.PUT_LINE(emp.salary);
  6      END LOOP;
  7  END;
  8  /
```

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

```
SQL> DECLARE
  2      i NUMBER := 1;
  3      emp_id NUMBER;
  4      emp_name VARCHAR2(20);
  5      salary NUMBER;
  6      department VARCHAR2(20);
  7  BEGIN
  8      WHILE i <= (SELECT COUNT(*) FROM employees) LOOP
  9          SELECT emp_id, emp_name, salary, department INTO emp_id, emp_name, salary, department
 10          FROM (SELECT emp_id, emp_name, salary, department, ROW_NUMBER() OVER (ORDER BY emp_id) AS row_num FROM employees)
 11          WHERE row_num = i;
 12
 13          FOR dup IN (SELECT emp_id FROM employees WHERE emp_name = emp_name AND salary = salary AND department = department AND emp_id <> emp_id
) LOOP
 14              DELETE FROM employees
 15              WHERE emp_id = dup.emp_id;
 16          END LOOP;
 17
 18          i := i + 1;
 19      END LOOP;
 20  END;
 21  /
```