Shivansh Lohani
23070521141

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 emplloyees (emp_id, emp_name, salary, department
  3      VALUES (101, 'John Doe', 5000, 'IT');
  4      COMMIT;
  5  END;
  6  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2      v_name emplloyees.emp_name%TYPE;
  3      CURSOR emp_cursor IS SELECT emp_name FROM emplloyees;
  4  BEGIN
  5      OPEN emp_cursor;
  6      LOOP
  7          FETCH emp_cursor INTO v_name;
  8          EXIT WHEN emp_cursor%NOTFOUND;
  9          DBMS_OUTPUT.PUT_LINE(v_name);
 10      END LOOP;
 11      CLOSE emp_cursor;
 12  END;
 13  /

PL/SQL procedure successfully completed.
```

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 emplloyees
  3       SET salary = salary * 1.10
  4       WHERE emp_id = 101;
  5       COMMIT;
  6  END;
  7  /

PL/SQL procedure successfully completed.
```

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

```
SQL> BEGIN
  2       DELETE FROM emplloyees WHERE emp_id = 105;
  3       COMMIT;
  4  END;
  5  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2       v_count NUMBER;
  3  BEGIN
  4       SELECT COUNT(*) INTO v_count FROM emplloyees;
  5       DBMS_OUTPUT.PUT_LINE('Total Employees: ' || v_count);
  6  END;
  7  /

PL/SQL procedure successfully completed.
```

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       v_salary NUMBER;
  3  BEGIN
  4       SELECT salary INTO v_salary FROM emplloyees WHERE emp_id = 101;
  5       IF v_salary > 5000 THEN
  6           DBMS_OUTPUT.PUT_LINE('High Salary');
  7       ELSE
  8           DBMS_OUTPUT.PUT_LINE('Low Salary');
  9       END IF;
 10  END;
 11  /

PL/SQL procedure successfully completed.
```

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      v_department emplloyees.department%TYPE;
  3  BEGIN
  4      SELECT department INTO v_department FROM emplloyees WHERE emp_id = 101;
  5      IF v_department = 'IT' THEN
  6          DBMS_OUTPUT.PUT_LINE('IT Department');
  7      ELSIF v_department = 'HR' THEN
  8          DBMS_OUTPUT.PUT_LINE('HR Department');
  9      ELSE
 10          DBMS_OUTPUT.PUT_LINE('Other Department');
 11      END IF;
 12  END;
 13  /

PL/SQL procedure successfully completed.
```

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      v_salary emplloyees.salary%TYPE;
  3      v_category VARCHAR2(20);
  4  BEGIN
  5      SELECT salary INTO v_salary FROM emplloyees WHERE emp_id = 101;
  6      v_category := CASE
  7          WHEN v_salary > 8000 THEN 'Senior Level'
  8          WHEN v_salary BETWEEN 5000 AND 8000 THEN 'Mid Level'
  9          ELSE 'Junior Level'
 10      END;
 11      DBMS_OUTPUT.PUT_LINE('Employee Category: ' || v_category);
 12  END;
 13  /

PL/SQL procedure successfully completed.
```

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

```
SQL> BEGIN
  2      UPDATE emplloyees
  3      SET salary = salary * 1.05
  4      WHERE department = 'Sales';
  5      COMMIT;
  6      DBMS_OUTPUT.PUT_LINE('Salary increased for Sales department!');
  7  END;
  8  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2      v_count NUMBER;
  3  BEGIN
  4      SELECT COUNT(*) INTO v_count FROM emplloyees WHERE emp_id = 110;
  5      IF v_count = 0 THEN
  6          INSERT INTO emplloyees (emp_id, emp_name, salary, department)
  7          VALUES (110, 'New Employee', 4500, 'HR');
  8          COMMIT;
  9          DBMS_OUTPUT.PUT_LINE('New employee inserted!');
 10      ELSE
 11          DBMS_OUTPUT.PUT_LINE('Employee already exists!');
 12      END IF;
 13  END;
 14  /

PL/SQL procedure successfully completed.
```

Tasks on Loops with Database

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

```
SQL> DECLARE
  2      v_name emplloyees.emp_name%TYPE;
  3  BEGIN
  4      FOR rec IN (SELECT emp_name FROM emplloyees) LOOP
  5          DBMS_OUTPUT.PUT_LINE('Employee Name: ' || rec.emp_name);
  6      END LOOP;
  7  END;
  8  /

PL/SQL procedure successfully completed.
```

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

```
SQL> BEGIN
  2      FOR i IN 1..5 LOOP
  3          INSERT INTO emplloyees (emp_id, emp_name, salary, department)
  4          VALUES (200 + i, 'Employee_' || i, 4000 + (i * 500), 'IT');
  5          DBMS_OUTPUT.PUT_LINE('Inserted Employee_' || i);
  6      END LOOP;
  7      COMMIT;
  8      DBMS_OUTPUT.PUT_LINE('5 Employees inserted successfully!');
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2      CURSOR emp_cursor IS SELECT emp_id, salary FROM emplloyees WHERE salary < 4000;
  3      v_emp_id emplloyees.emp_id%TYPE;
  4      v_salary emplloyees.salary%TYPE;
  5  BEGIN
  6      OPEN emp_cursor;
  7      LOOP
  8          FETCH emp_cursor INTO v_emp_id, v_salary;
  9          EXIT WHEN emp_cursor%NOTFOUND;
 10
 11          UPDATE emplloyees
 12          SET salary = salary * 1.20
 13          WHERE emp_id = v_emp_id;
 14
 15          DBMS_OUTPUT.PUT_LINE('Salary increased for Employee ID: ' || v_emp_id);
 16      END LOOP;
 17      CLOSE emp_cursor;
 18      COMMIT;
 19      DBMS_OUTPUT.PUT_LINE('Salary updated for all applicable employees.');
 20  END;
 21  /

PL/SQL procedure successfully completed.
```

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).

```
SQL> BEGIN
  2      DELETE FROM emplloyees
  3      WHERE last_updated < ADD_MONTHS(SYSDATE, -60);
  4
  5      DBMS_OUTPUT.PUT_LINE('Deleted employees with outdated records.');
  6      COMMIT;
  7  END;
  8  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2      v_name emplloyees.emp_name%TYPE;
  3      v_salary emplloyees.salary%TYPE;
  4  BEGIN
  5      SELECT emp_name, salary INTO v_name, v_salary FROM emplloyees
  6      WHERE salary = (SELECT MAX(salary) FROM.emplloyees);
  7
  8      DBMS_OUTPUT.PUT_LINE('Highest Paid: ' || v_name || ' with salary ' || v_salary);
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2      v_dept emplloyees.department%TYPE := 'IT';  -- Change as needed
  3      CURSOR dept_cursor IS SELECT emp_name FROM emplloyees WHERE department = v_dept;
  4      v_emp_name emplloyees.emp_name%TYPE;
  5  BEGIN
  6      OPEN dept_cursor;
  7      LOOP
  8          FETCH dept_cursor INTO v_emp_name;
  9          EXIT WHEN dept_cursor%NOTFOUND;
 10          DBMS_OUTPUT.PUT_LINE('Employee: ' || v_emp_name);
 11      END LOOP;
 12      CLOSE dept_cursor;
 13  END;
 14  /

PL/SQL procedure successfully completed.
```

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.

```
SQL> DECLARE
  2      CURSOR high_salary_cursor IS
  3          SELECT emp_name, salary FROM emplloyees ORDER BY salary DESC FETCH FIRST 5 ROWS ONLY;
  4  BEGIN
  5      FOR rec IN high_salary_cursor LOOP
  6          DBMS_OUTPUT.PUT_LINE('Employee: ' || rec.emp_name || ', Salary: ' || rec.salary);
  7      END LOOP;
  8  END;
  9  /

PL/SQL procedure successfully completed.
```

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

```
SQL> DECLARE
  2      CURSOR dup_cursor IS
  3          SELECT emp_id FROM emplloyees
  4          GROUP BY emp_id
  5          HAVING COUNT(emp_id) > 1;
  6      v_emp_id emplloyees.emp_id%TYPE;
  7  BEGIN
  8      OPEN dup_cursor;
  9      LOOP
 10          FETCH dup_cursor INTO v_emp_id;
 11          EXIT WHEN dup_cursor%NOTFOUND;
 12
 13          DELETE FROM emplloyees WHERE emp_id = v_emp_id
 14          AND ROWID NOT IN (SELECT MIN(ROWID) FROM emplloyees WHERE emp_id = v_emp_id);
 15
 16          DBMS_OUTPUT.PUT_LINE('Deleted duplicate for Employee ID: ' || v_emp_id);
 17      END LOOP;
 18      CLOSE dup_cursor;
 19      COMMIT;
 20      DBMS_OUTPUT.PUT_LINE('All duplicate records removed.');
 21  END;
 22  /

PL/SQL procedure successfully completed.
```