Database and Management System Lab

Lab Experiment – 14

Name: Kartikeya Singh

Roll No: R2142230051

Sap_ID: 500123812

B.Tech CSE, SEM-III, B-2

Title: To understand the concepts of function and procedure in PL/SQL.

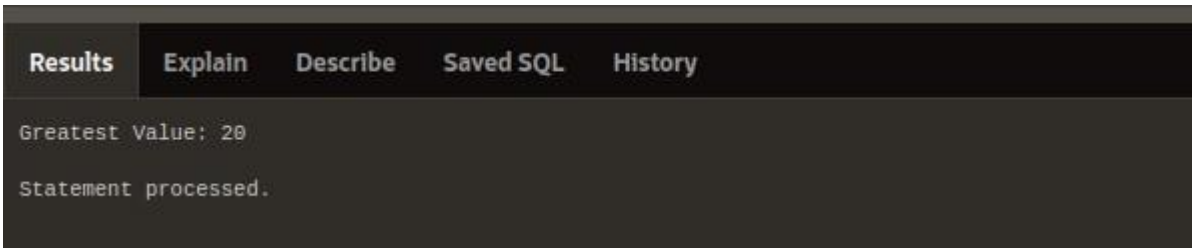
Objective: Students will be able to implement the PI/SQL programs using function and procedure.

Implement the above experiments of PL/SQL using functions and procedures.

1. Function to Find the Greatest Value

```
CREATE OR REPLACE FUNCTION find_greatest (
  a IN NUMBER,
  b IN NUMBER,
  c IN NUMBER
)
RETURN NUMBER
AS
  greatest_value NUMBER;
BEGIN
  greatest_value := GREATEST(a, b, c);
  RETURN greatest_value;
END;
/
```

```
BEGIN
  DBMS_OUTPUT.PUT_LINE(
    'Greatest Value: ' || find_greatest(10, 20, 15)
  );
END;
/
```



2. Procedure to Display a Welcome Message 20 Times

```
CREATE OR REPLACE PROCEDURE display_welcome
AS
BEGIN
  FOR i IN 1..20 LOOP
    DBMS_OUTPUT.PUT_LINE('Welcome to PL/SQL Programming');
  END LOOP;
END;
/
```

```sql
BEGIN
  display_welcome;
END;
/
```

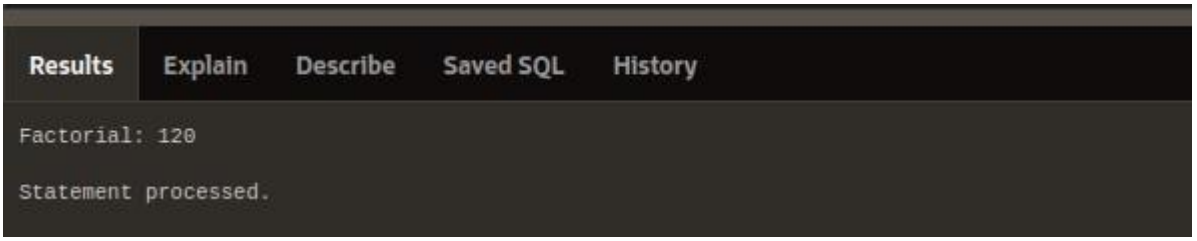
Results Explain Describe Saved SQL History

```
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
```

3. Function to Find the Factorial of a Number

```sql
CREATE OR REPLACE FUNCTION find_factorial (
  n IN NUMBER
)
RETURN NUMBER
AS
  factorial NUMBER := 1;
BEGIN
  FOR i IN 1..n LOOP
    factorial := factorial * i;
  END LOOP;
  RETURN factorial;
END;
/
```

```sql
BEGIN
  DBMS_OUTPUT.PUT_LINE('Factorial: ' || find_factorial(5));
END;
/
```

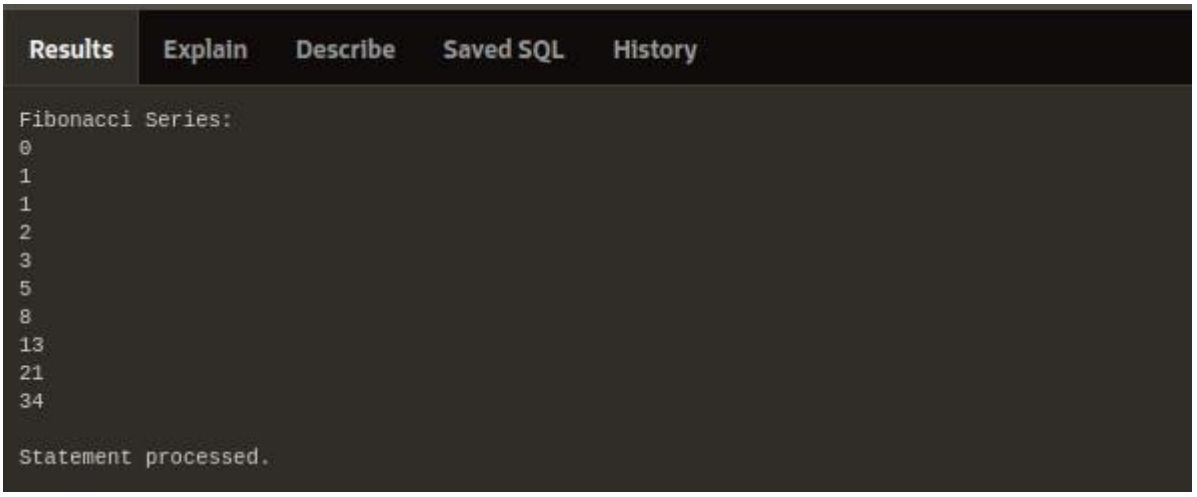


4. Procedure to Generate Fibonacci Series

```sql
CREATE OR REPLACE PROCEDURE generate_fibonacci (
  n IN NUMBER
)
AS
  a NUMBER := 0;
  b NUMBER := 1;
  temp NUMBER;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');
  DBMS_OUTPUT.PUT_LINE(a);
  DBMS_OUTPUT.PUT_LINE(b);

  FOR i IN 3..n LOOP
    temp := a + b;
    DBMS_OUTPUT.PUT_LINE(temp);
    a := b;
    b := temp;
  END LOOP;
END;
/
```

```sql
BEGIN
  generate_fibonacci(10);
END;
/
```



5. Function to Find the Sum of First N Numbers

```
CREATE OR REPLACE FUNCTION find_sum (
  n IN NUMBER
)
RETURN NUMBER
AS
  total NUMBER := 0;
BEGIN
  FOR i IN 1..n LOOP
    total := total + i;
  END LOOP;
  RETURN total;
END;
/
```

```
BEGIN
  DBMS_OUTPUT.PUT_LINE(
    'Sum of first 10 numbers: ' || find_sum(10)
  );
END;
/
```

```
Sum of first 10 numbers: 55

Statement processed.
```

Database and Management System Lab

Lab Experiment – 15

Name: Kartikeya Singh

Roll No: R2142230051

Sap_ID: 500123812

B.Tech CSE, SEM-III, B-2

Title: To understand the concepts of implicit and explicit cursor.

Objective: Students will be able to implement the concept of implicit and explicit cursor.

1. Implicit Cursor to update salaries

```
DECLARE
    v_count NUMBER;
BEGIN
    UPDATE EMPLOYEES
    SET SALARY = SALARY * 1.1;

    v_count := SQL%ROWCOUNT;

    IF v_count > 0 THEN
        DBMS_OUTPUT.PUT_LINE(
            v_count || ' records have been updated.'
        );
    ELSE
        DBMS_OUTPUT.PUT_LINE('No Change');
    END IF;
    COMMIT;
END;
/
```

2. Explicit cursor to fetch employee detail

```
DECLARE
    CURSOR emp_cursor IS
        SELECT employee_id, first_name, last_name, salary
        FROM EMPLOYEES;
    emp_record emp_cursor%ROWTYPE;
BEGIN
    OPEN emp_cursor;
    LOOP
        FETCH emp_cursor INTO emp_record;
        EXIT WHEN emp_cursor%NOTFOUND;
        DBMS_OUTPUT.PUT_LINE('Employee ID: ' || emp_record.employee_id ||
                             ', Name: ' || emp_record.first_name || ' ' ||
                            emp_record.last_name ||
                            ', Salary: ' || emp_record.salary);
    END LOOP;
    CLOSE emp_cursor;
END;
/
```

3. Explicit Cursor to insert high salary records into TEMP_EMP

```
1   DECLARE
2       CURSOR high_salary_cursor IS
3           SELECT employee_id, last_name, salary
4           FROM EMPLOYEES
5           WHERE salary > 2500;
6   BEGIN
7       DELETE FROM TEMP_EMP;
8
9       FOR emp_record IN high_salary_cursor LOOP
10          INSERT INTO TEMP_EMP (employee_id, last_name, salary)
11          VALUES (emp_record.employee_id, emp_record.last_name, emp_record.sal
12      END LOOP;
13
14      COMMIT;
15      DBMS_OUTPUT.PUT_LINE('High salary records inserted into TEMP_EMP');
16  END;
17  /
```

Results Explain Describe Saved SQL History

```
High salary records inserted into TEMP_EMP

1 row(s) inserted.
```

```
1   SELECT * FROM TEMP_EMP;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	LAST_NAME	SALARY
2	Smith	3080
3	Johnson	3300
4	Williams	2530
5	Brown	3850