DEPARTMENT OF COMPUTER SYSTEMS ENGINEERING
MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO
Database Management Systems (4th Semester) 18CS
Lab Experiment 14

Roll No: **Date of Conduct:**
SubmissionDate: **Grade Obtained:**

Problem Recognition (0.3)	Completeness & accuracy (0.4)	Timeliness (0.3)	Score (1.0)

Objective: Working with stored Functions and Procedures in PL/SQL.

Tools: MySql/oracle.

Introduction:

Stored Functions and Procedures in PL/SQL

Stored procedures and functions (subprograms) can be compiled and stored in an Oracle Database, ready to be executed. Once compiled, it is a schema object known as a stored procedure or stored function, which can be referenced or called any number of times by multiple applications connected to Oracle Database. Both stored procedures and functions can accept parameters when they are executed (called). To execute a stored procedure or function, you only need to include its object name.

Creating a Stored Procedure or Function

Whenever a block of code for stored procedure or function is written it is then, they are automatically compiled by the oracle engine. During compilation if any error occurs, we get a message on the screen saying that the procedure or function is created with compilation errors but actual error is not displayed.

In order to find out the compilation errors following statement can be executed:

```
SELECT * from USER_ERRORS;
```

Once it is compiled, it is then stored by the oracle engine in the database as a database object. Stored Procedure or function's block of code in PL/SQL is made up of the following three sections:

- **Declarative section:** In this section, variables, constants, cursor or exceptions that are going to be used by procedure or function are declared.

- **Executable section:** In this section, the definition of procedure or function created is written. This section also contains the SQL or PL/SQL statements assigning values, controlling execution and manipulating data.
- **Exception Handling section:** In this section, the expected exceptions are written which may arise during execution of code written in executable part. This section is optional.

Syntax for creating Stored Procedure

Below we have the basic syntax for creating a stored procedure in oracle:

```
CREATE OR REPLACE PROCEDURE <procedure_name>

(<variable_name>IN/OUT/IN OUT <datatype>,

  <variable_name>IN/OUT/IN OUT <datatype>,...) IS/AS

variable/constant declaration;

BEGIN

          -- PL/SQL subprogram body;

EXCEPTION

          -- Exception Handling block ;

END <procedure_name>;
```

Let's understand the above code,

- procedure_name is for procedure's name and variable_name is the variable name for variable used in the stored procedure.
- CREATE or REPLACE PROCEDURE is a keyword used for specifying the name of the procedure to be created.
- BEGIN, EXCEPTION and END are keywords used to indicate different sections of the procedure being created.
- IN/OUT/IN OUT are parameter modes.
- IN mode refers to READ ONLY mode which is used for a variable by which it will accept the value from the user. It is the default parameter mode.
- OUT mode refers to WRITE ONLY mode which is used for a variable that will return the value to the user.
- IN OUT mode refers to READ AND WRITE mode which is used for a variable that will either accept a value from the user or it will return the value to the user.
- At the end, <procedure_name> is optional to write, you can simply use END statement to end the procedure definition.

Syntax for creating Functions in PL/SQL
Now that we know how to create stored procedures and how to use them in another PL/SQL code block, it's time to understand how to create Functions in PL/SQL.

```
CREATE OR REPLACE FUNCTION <function_name>

(<variable_name> IN <datatype>,

<variable_name> IN <datatype>,...)

RETURN <datatype> IS/AS

variable/constant declaration;

BEGIN

        -- PL/SQL subprogram body;

EXCEPTION

        -- Exception Handling block ;

END <function_name>;
```

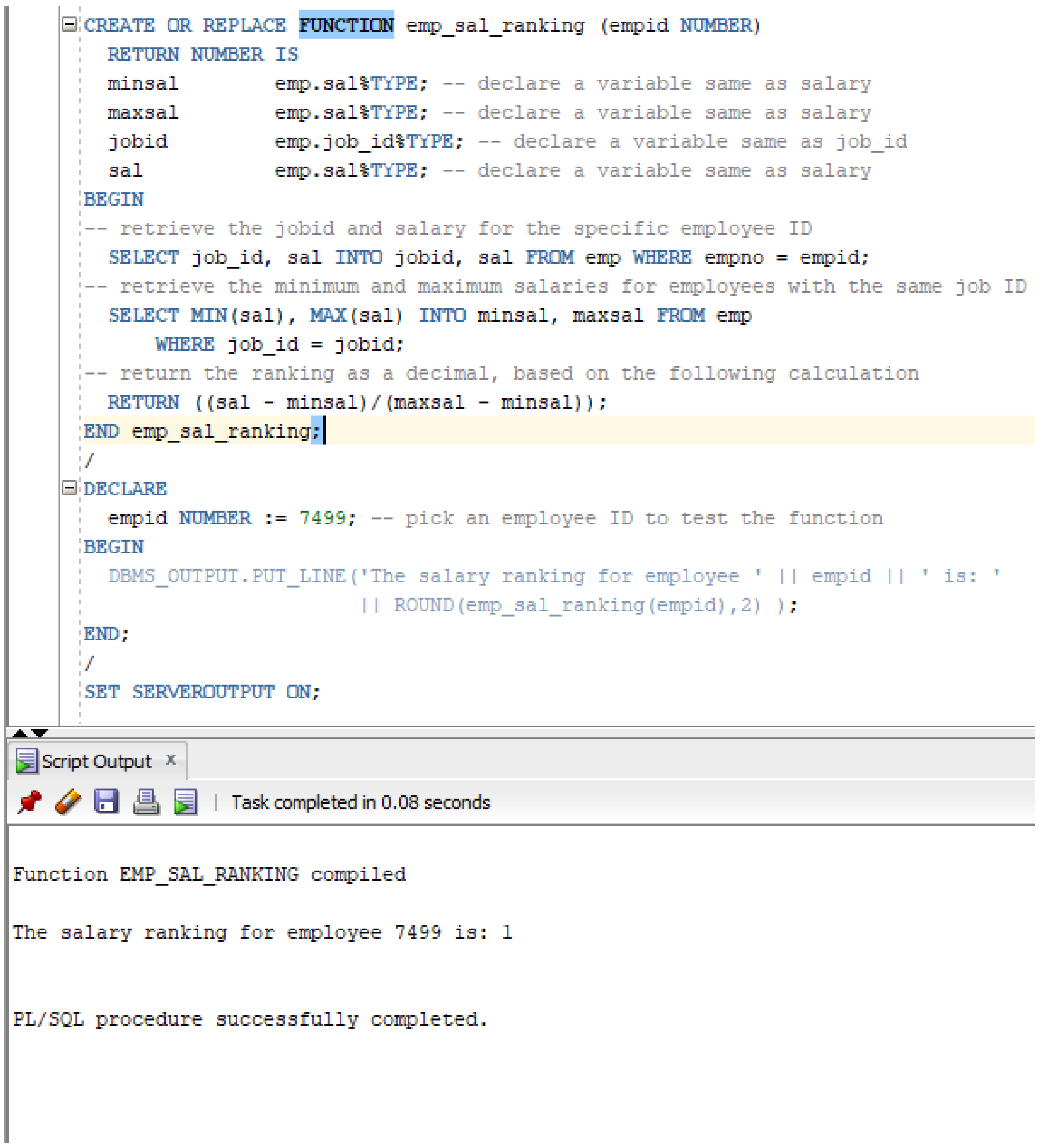
Let's understand the above code:

- Function_name is for defining function's name and variable_name is the variable name for variable used in the function.
- CREATE or REPLACE FUNCTION is a keyword used for specifying the name of the function to be created.
- IN mode refers to READ ONLY mode which is used for a variable by which it will accept the value from the user. It is the default parameter mode.
- RETURN is a keyword followed by a datatype specifying the datatype of a value that the function will return.

LAB TASK

1. Create a stored function to calculate the salary ranking of the employee based on the current minimum and maximum salaries for employees in the same job category. (Hint: salary ranking = (empsal-minsal)/(Maxsal-Minsal)).

Task:



```
CREATE OR REPLACE FUNCTION emp_sal_ranking (empid NUMBER)
  RETURN NUMBER IS
  minsal          emp.sal%TYPE; -- declare a variable same as salary
  maxsal          emp.sal%TYPE; -- declare a variable same as salary
  jobid           emp.job_id%TYPE; -- declare a variable same as job_id
  sal             emp.sal%TYPE; -- declare a variable same as salary
BEGIN
-- retrieve the jobid and salary for the specific employee ID
  SELECT job_id, sal INTO jobid, sal FROM emp WHERE empno = empid;
-- retrieve the minimum and maximum salaries for employees with the same job ID
  SELECT MIN(sal), MAX(sal) INTO minsal, maxsal FROM emp
      WHERE job_id = jobid;
-- return the ranking as a decimal, based on the following calculation
  RETURN ((sal - minsal)/(maxsal - minsal));
END emp_sal_ranking;
/
DECLARE
  empid NUMBER := 7499; -- pick an employee ID to test the function
BEGIN
  DBMS_OUTPUT.PUT_LINE('The salary ranking for employee ' || empid || ' is: '
                      || ROUND(emp_sal_ranking(empid),2) );
END;
/
SET SERVEROUTPUT ON;
```

Script Output

Task completed in 0.08 seconds

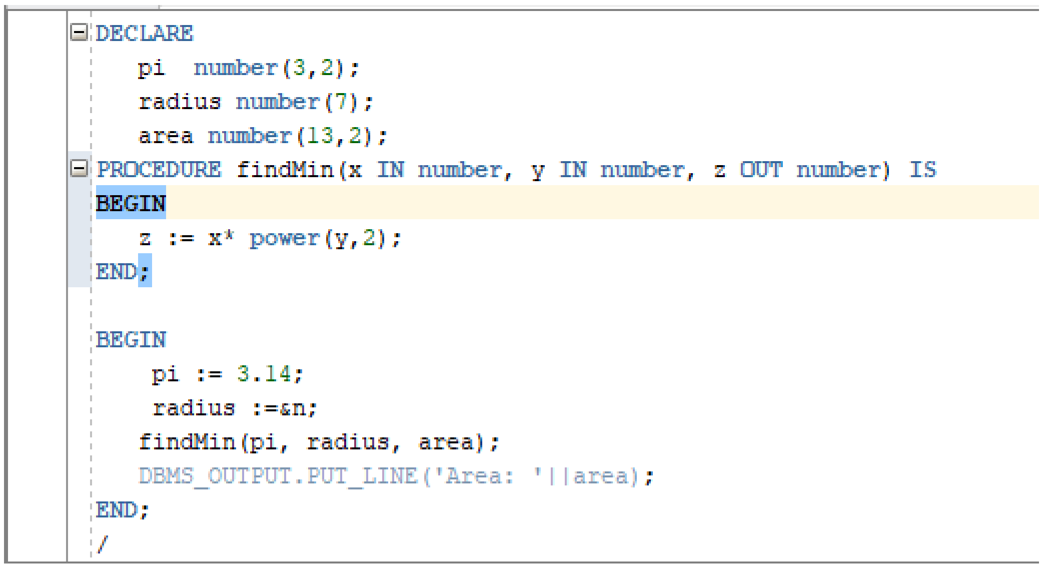
```
Function EMP_SAL_RANKING compiled

The salary ranking for employee 7499 is: 1


PL/SQL procedure successfully completed.
```

2. Write a stored procedure that accepts radius of circle and displays area of the circle.

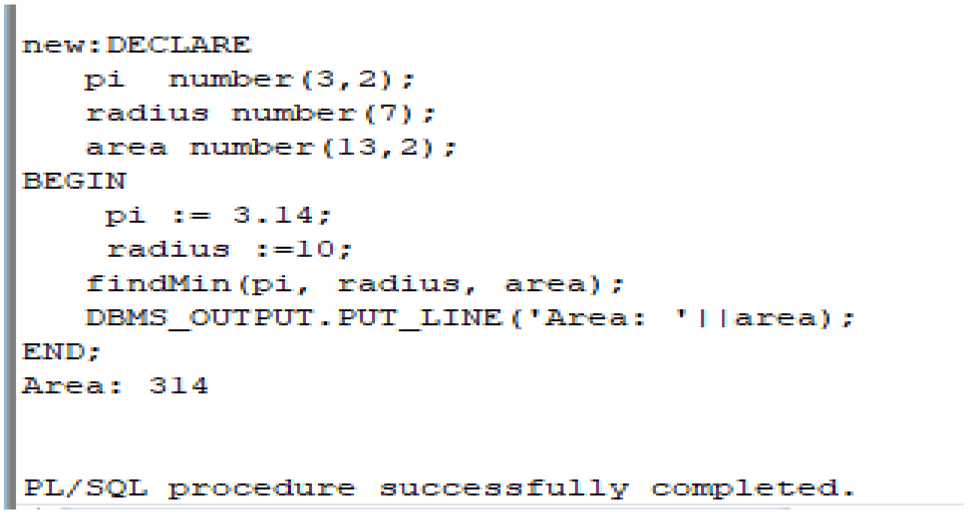
Task:



```
DECLARE
   pi  number(3,2);
   radius number(7);
   area number(13,2);
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
   z := x* power(y,2);
END;

BEGIN
    pi := 3.14;
    radius :=&n;
   findMin(pi, radius, area);
   DBMS_OUTPUT.PUT_LINE('Area: '||area);
END;
/
```

Output:

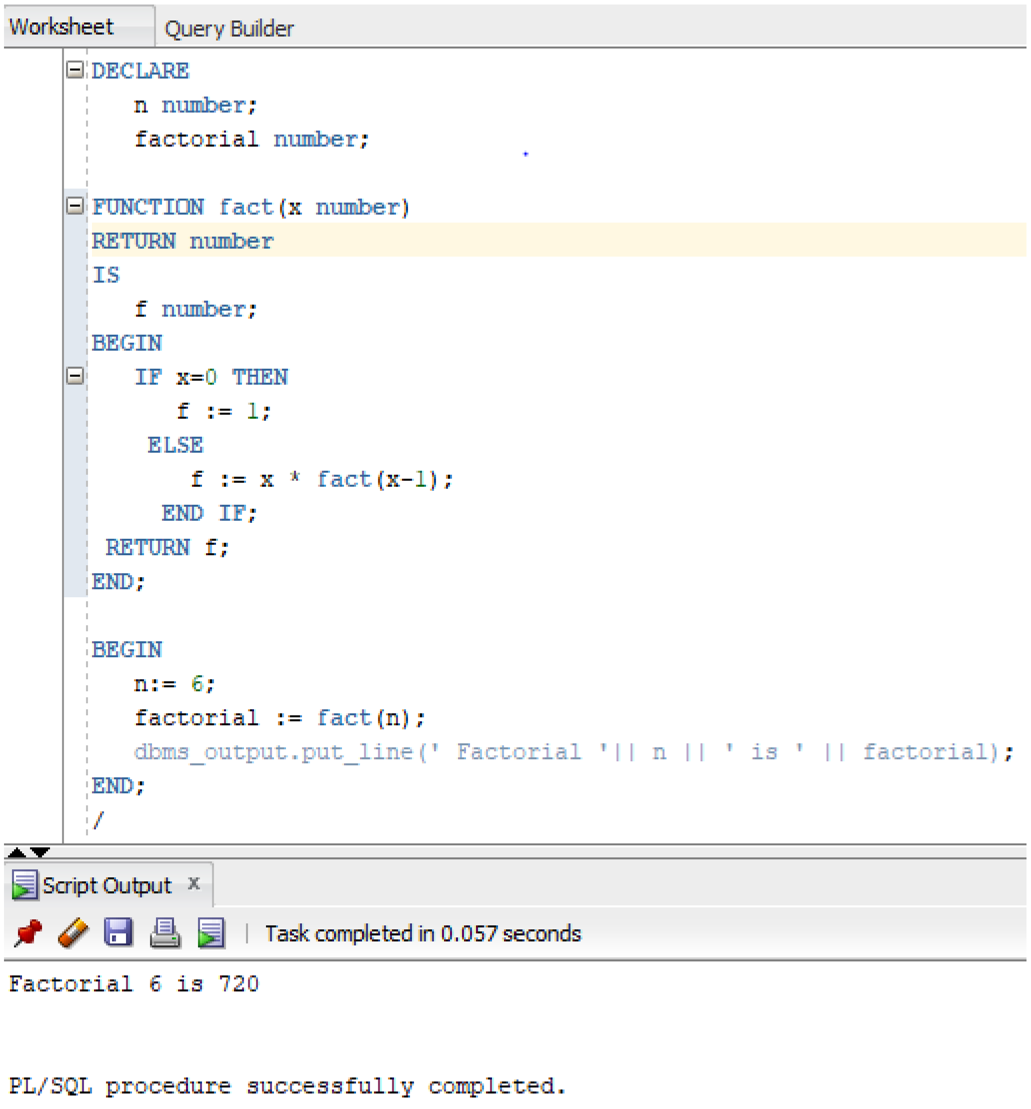


```
new:DECLARE
   pi  number(3,2);
   radius number(7);
   area number(13,2);
BEGIN
    pi := 3.14;
    radius :=10;
   findMin(pi, radius, area);
   DBMS_OUTPUT.PUT_LINE('Area: '||area);
END;
Area: 314


PL/SQL procedure successfully completed.
```

3. Write a stored function that accepts a number and returns its factorial.

Task:



```
DECLARE
   n number;
   factorial number;

FUNCTION fact(x number)
RETURN number
IS
   f number;
BEGIN
   IF x=0 THEN
      f := 1;
    ELSE
       f := x * fact(x-1);
     END IF;
 RETURN f;
END;

BEGIN
   n:= 6;
   factorial := fact(n);
   dbms_output.put_line(' Factorial '|| n || ' is ' || factorial);
END;
/
```

```
Factorial 6 is 720


PL/SQL procedure successfully completed.
```