Ex No : 9 **Procedures, Functions and packages in PL/SQL**

Date : 25-03-2024

Aim:

To execute the functions, procedures and Packages in SQL

Description

Functions:

A function is a named PL/SQL Block which is similar to a procedure. The major difference
between a procedure and a function is, a function must always return a value, but a procedure
may or may not return a value.

SYNTAX

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
RETURN return_datatype;
IS
Declaration_section
BEGIN
Execution_section
Return return_variable;
EXCEPTION
exception section
Return return_variable;
END;
```

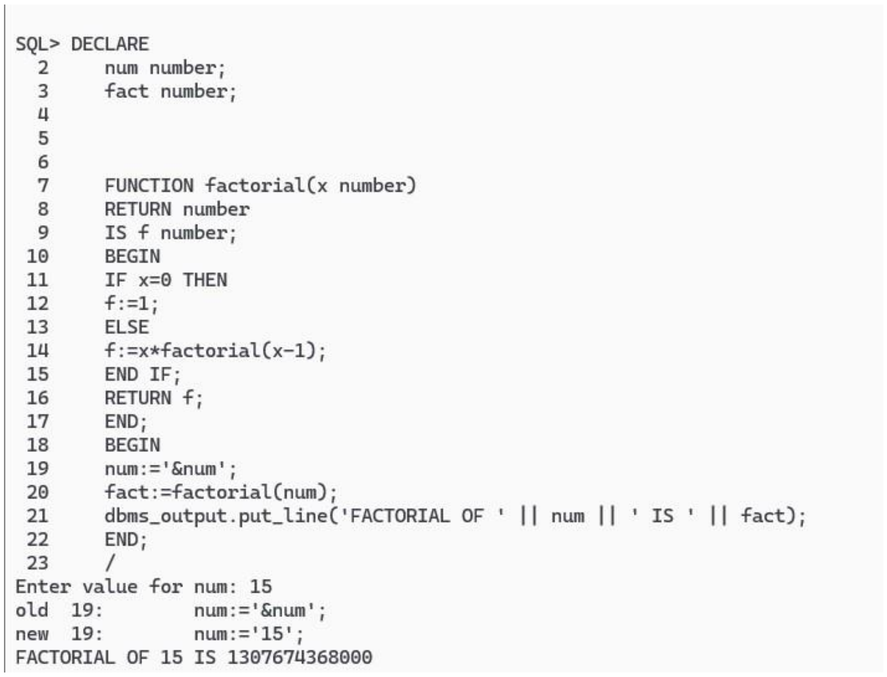
Sample Programs:

Factorial of a number:

```
SQL> DECLARE
  1  num number;
  2  fact number;
```

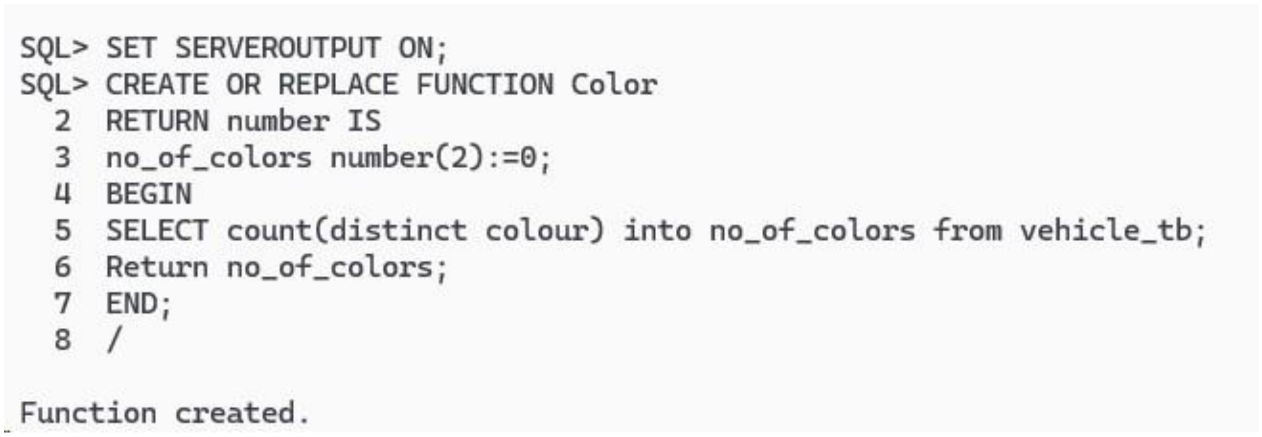
```
FUNCTION factorial(x number)
RETURN number
IS f number;
BEGIN
IF x=0 THEN
f:=1;
ELSE
f:=x*factorial(x-1);
END IF;
RETURN f;
END;
BEGIN
num:='&num';
fact:=factorial(num);
dbms_output.put_line('FACTORIAL OF ' || num || ' IS ' || fact);
END;
/
```

OUTPUT:

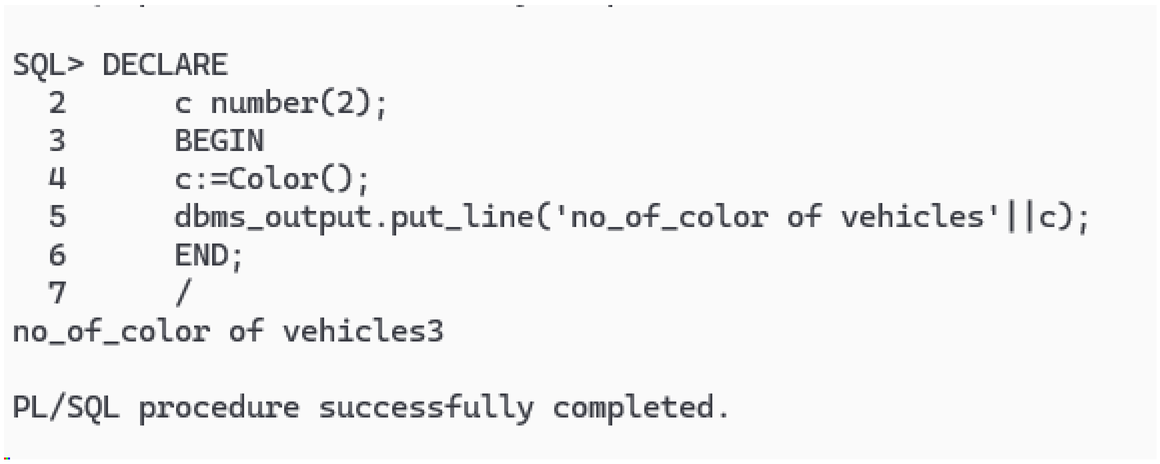


```
SQL> DECLARE
  2     num number;
  3     fact number;
  4
  5
  6
  7     FUNCTION factorial(x number)
  8     RETURN number
  9     IS f number;
 10     BEGIN
 11     IF x=0 THEN
 12     f:=1;
 13     ELSE
 14     f:=x*factorial(x-1);
 15     END IF;
 16     RETURN f;
 17     END;
 18     BEGIN
 19     num:='&num';
 20     fact:=factorial(num);
 21     dbms_output.put_line('FACTORIAL OF ' || num || ' IS ' || fact);
 22     END;
 23     /
Enter value for num: 15
old  19:        num:='&num';
new  19:        num:='15';
FACTORIAL OF 15 IS 1307674368000
```

Application:



```
SQL> SET SERVEROUTPUT ON;
SQL> CREATE OR REPLACE FUNCTION Color
RETURN number IS
no_of_colors number(2):=0;
BEGIN
SELECT count(distinct colour) into no_of_colors from vehicle_tb;
Return no_of_colors;
END;
/

Function created.
```



```
SQL> DECLARE
     c number(2);
     BEGIN
     c:=Color();
     dbms_output.put_line('no_of_color of vehicles'||c);
     END;
     /
no_of_color of vehicles3

PL/SQL procedure successfully completed.
```

Procedure:

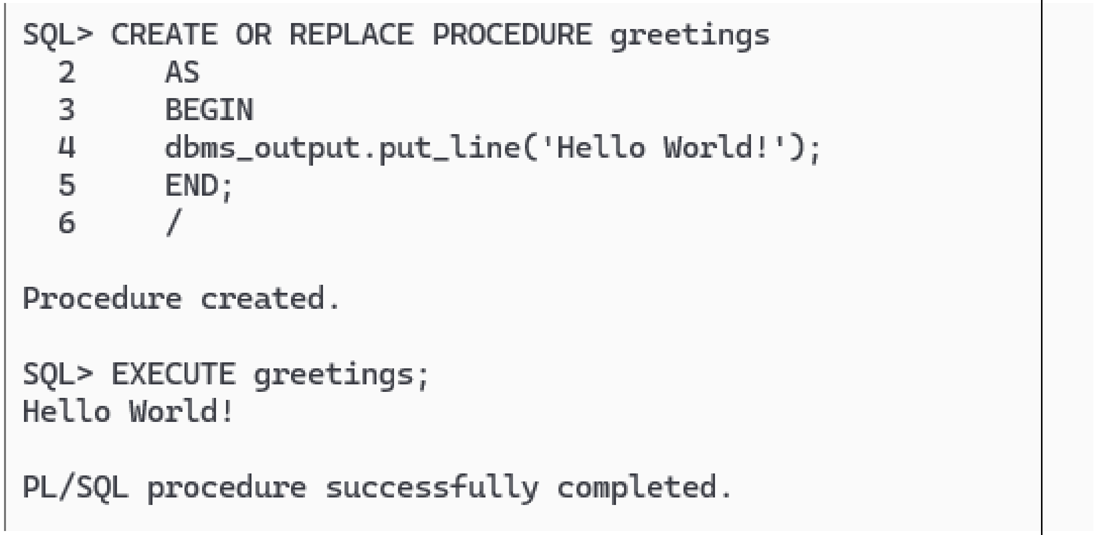
- A **procedure** is a group of **PL/SQL** statements that you can call by name.
- A procedure is a module performing one or more actions , it does not need to return any values.

```
CREATE [OR REPLACE] PROCEDURE
procedure_name [(parameter_name
[IN | OUT | IN OUT] type [, ...])]
```

Sample Programs:

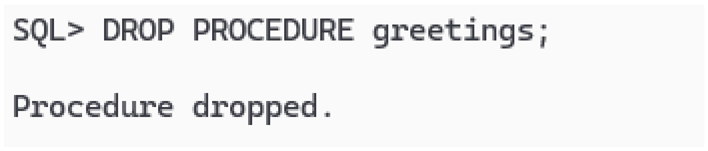
Creation and dropping of procedure:

```
SQL> CREATE OR REPLACE PROCEDURE greetings
1 AS
2  BEGIN
3  dbms_output.put_line('Hello World!');
4  END;
5 /
Procedure created.
```



```
SQL> CREATE OR REPLACE PROCEDURE greetings
  2     AS
  3     BEGIN
  4     dbms_output.put_line('Hello World!');
  5     END;
  6     /

Procedure created.

SQL> EXECUTE greetings;
Hello World!

PL/SQL procedure successfully completed.
```

SQL> DROP PROCEDURE greetings;

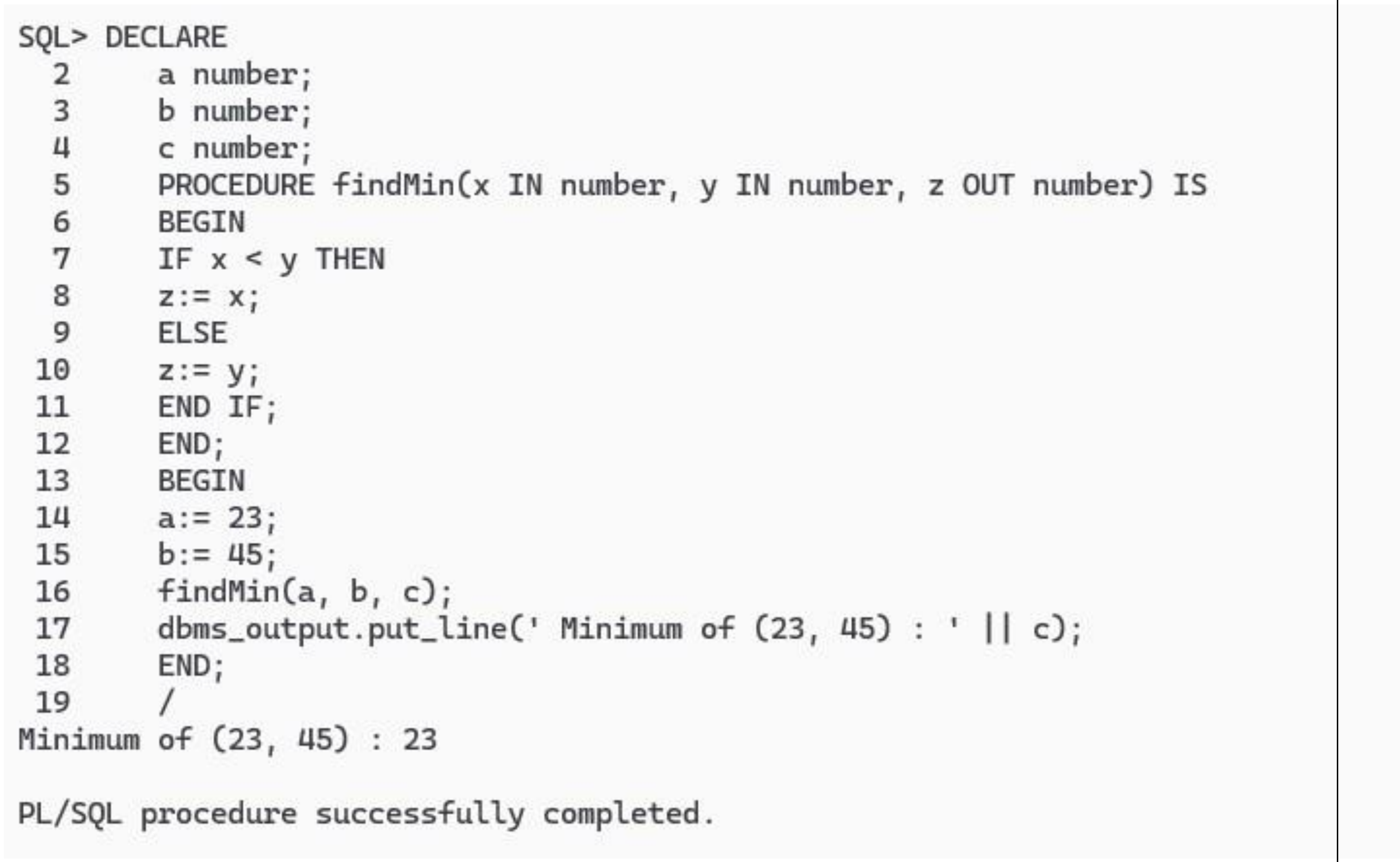


```
SQL> DROP PROCEDURE greetings;

Procedure dropped.
```

```
SQL> DECLARE
a number;
b number;
c number;
PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
BEGIN
IF x < y THEN
z:= x;
```

```
ELSE
z:= y;
END IF;
END;
BEGIN
a:= 23;
b:= 45;
findMin(a, b, c);
dbms_output.put_line(' Minimum of (23, 45) : ' || c);
END;
/
```

OUTPUT:

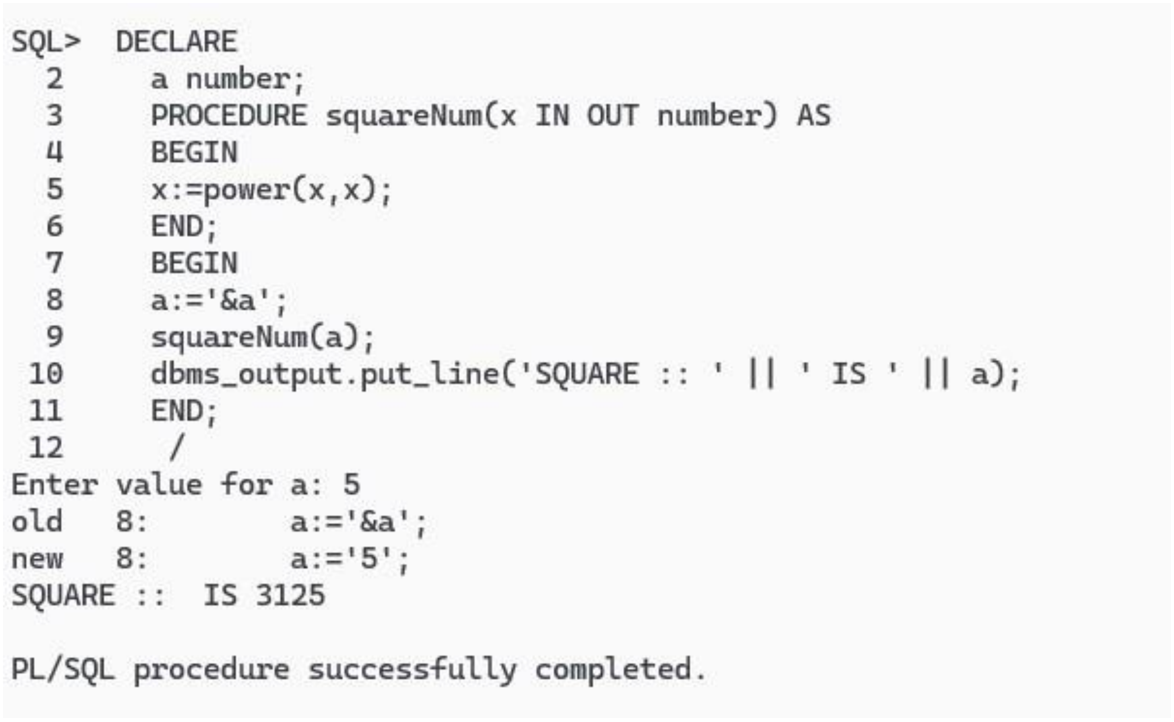


```
SQL> DECLARE
     a number;
     b number;
     c number;
     PROCEDURE findMin(x IN number, y IN number, z OUT number) IS
     BEGIN
     IF x < y THEN
     z:= x;
     ELSE
     z:= y;
     END IF;
     END;
     BEGIN
     a:= 23;
     b:= 45;
     findMin(a, b, c);
     dbms_output.put_line(' Minimum of (23, 45) : ' || c);
     END;
     /
Minimum of (23, 45) : 23

PL/SQL procedure successfully completed.
```

Square of a number

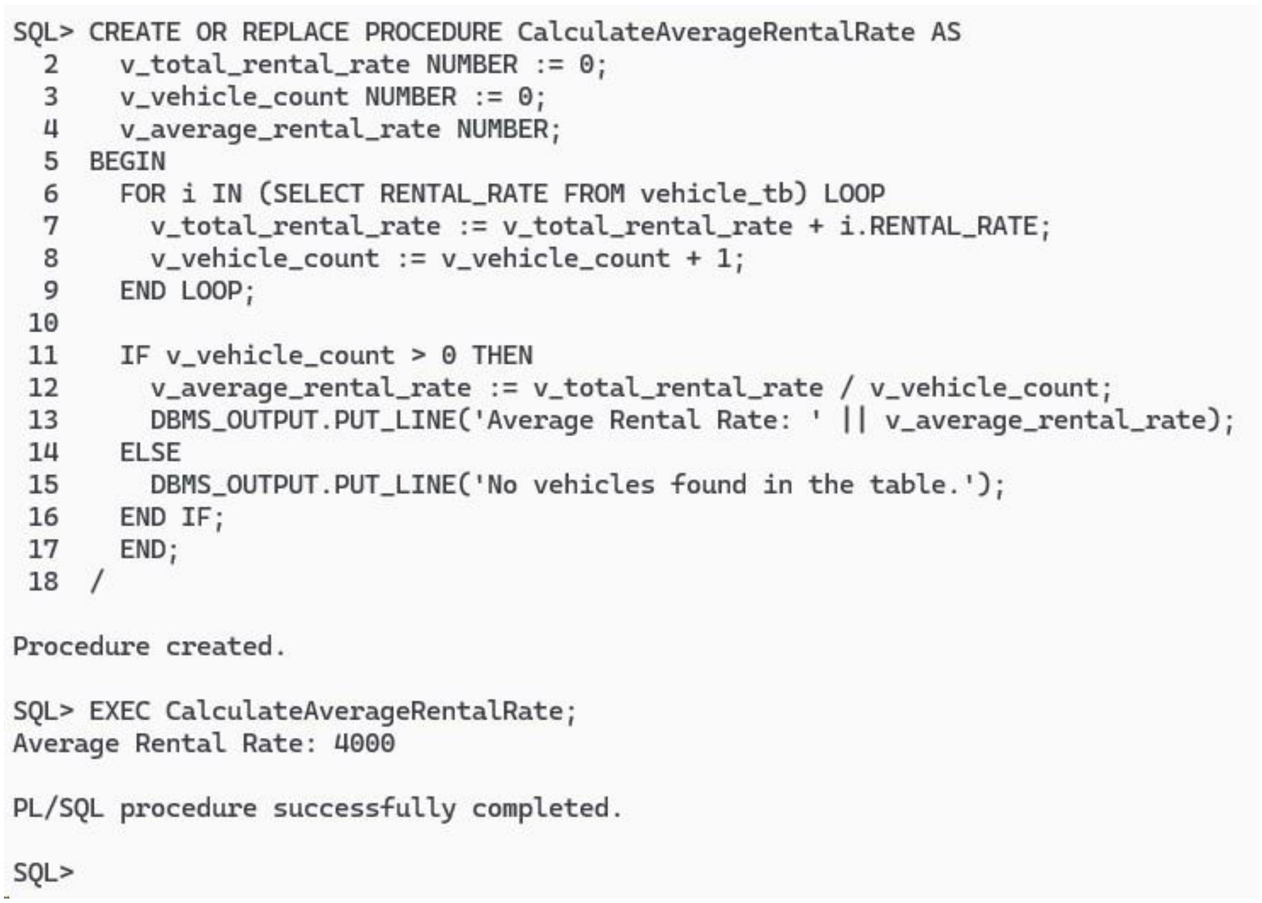
```
SQL> DECLARE
a number;
PROCEDURE squareNum(x IN OUT number) AS
BEGIN
x:=power(x,x);
END;
BEGIN
a:='&a';
squareNum(a);
dbms_output.put_line('SQUARE :: ' || ' IS ' || a);
END;
```

OUTPUT:



```
SQL>  DECLARE
        a number;
        PROCEDURE squareNum(x IN OUT number) AS
        BEGIN
        x:=power(x,x);
        END;
        BEGIN
        a:='&a';
        squareNum(a);
        dbms_output.put_line('SQUARE :: ' || ' IS ' || a);
        END;
         /
Enter value for a: 5
old   8:        a:='&a';
new   8:        a:='5';
SQUARE ::  IS 3125

PL/SQL procedure successfully completed.
```

MY APPLICATION:


```
SQL> CREATE OR REPLACE PROCEDURE CalculateAverageRentalRate AS
    v_total_rental_rate NUMBER := 0;
    v_vehicle_count NUMBER := 0;
    v_average_rental_rate NUMBER;
  BEGIN
    FOR i IN (SELECT RENTAL_RATE FROM vehicle_tb) LOOP
      v_total_rental_rate := v_total_rental_rate + i.RENTAL_RATE;
      v_vehicle_count := v_vehicle_count + 1;
    END LOOP;

    IF v_vehicle_count > 0 THEN
      v_average_rental_rate := v_total_rental_rate / v_vehicle_count;
      DBMS_OUTPUT.PUT_LINE('Average Rental Rate: ' || v_average_rental_rate);
    ELSE
      DBMS_OUTPUT.PUT_LINE('No vehicles found in the table.');
    END IF;
    END;
  /

Procedure created.

SQL> EXEC CalculateAverageRentalRate;
Average Rental Rate: 4000

PL/SQL procedure successfully completed.

SQL>
```

<u>PACKAGES:</u>

Packages are schema objects that groups logically related PL/SQL types, variables, and subprograms.

A package will have two mandatory parts –

- Package specification
- Package body or definition .

SYNTAX:

CREATE [OR REPLACE] PROCEDURE procedure_name

[(parameter_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

&lt; procedure_body &gt;

END

procedure_name;

For creating package body:

CREATE [OR REPLACE] PACKAGE BODY <package_name> IS

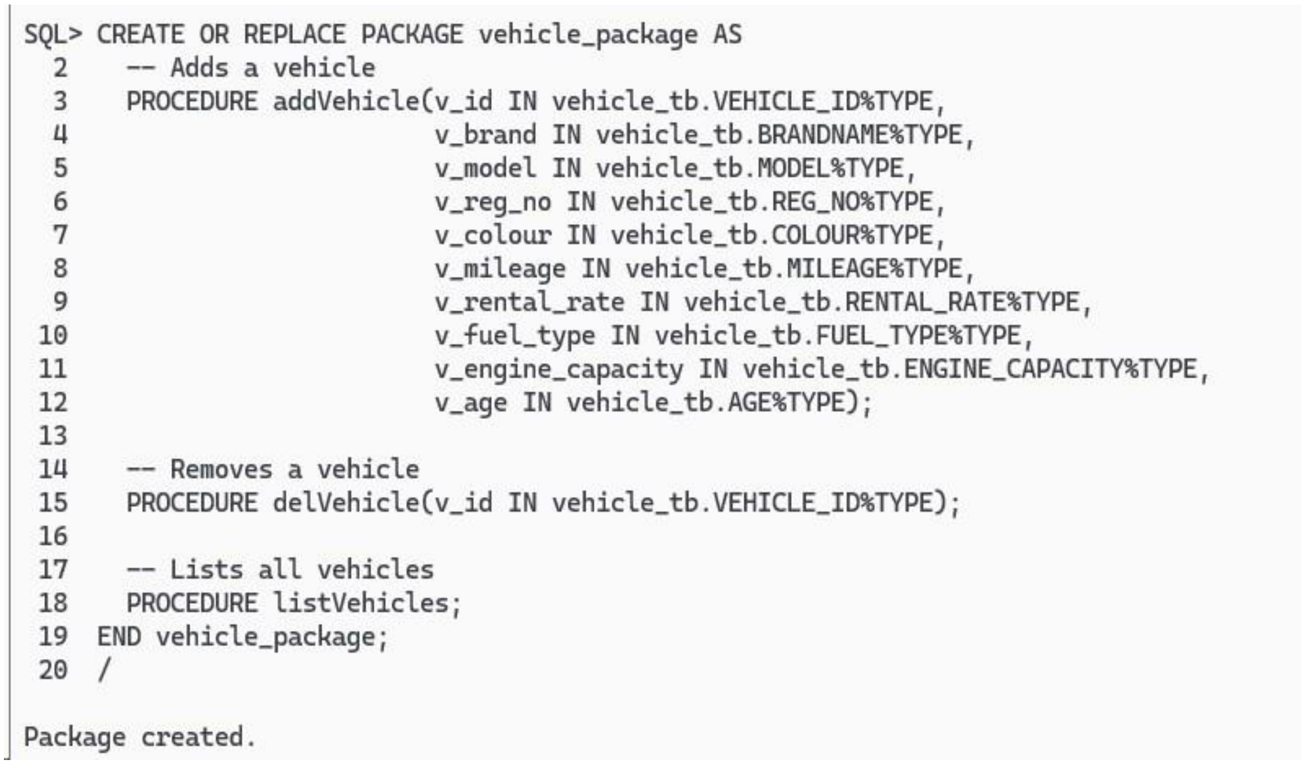
<global_declaration part>

<Private element definition>

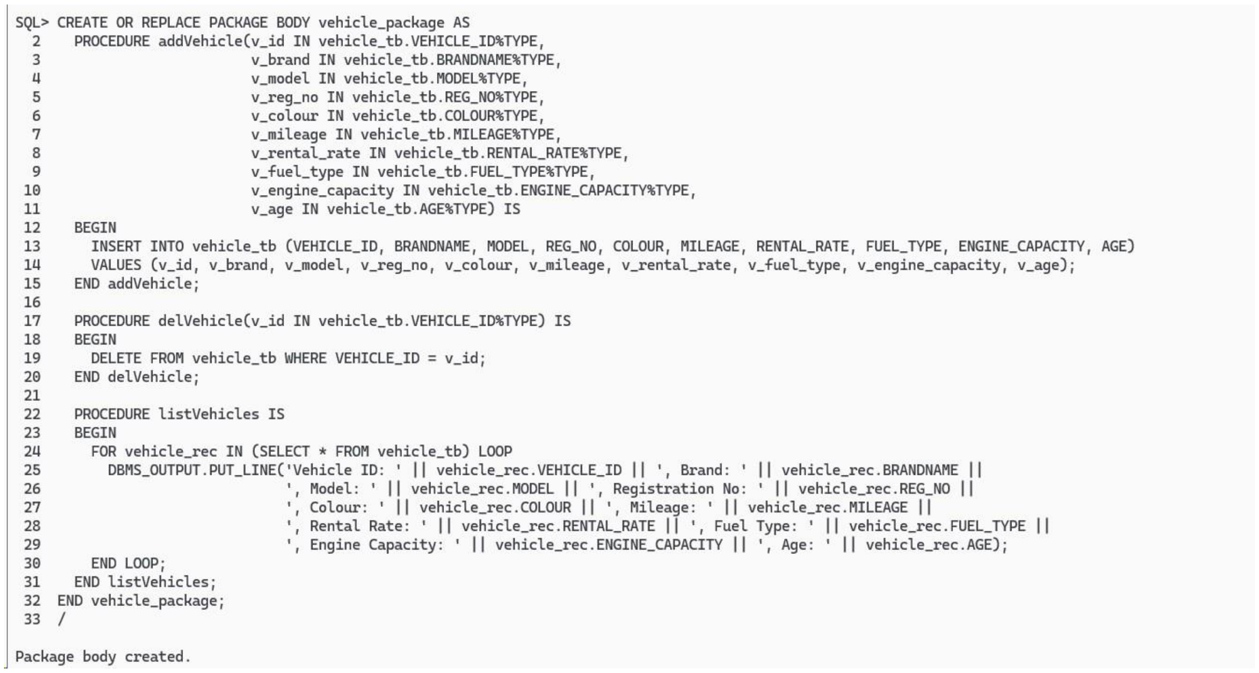
<sub_program and public element definition>

.

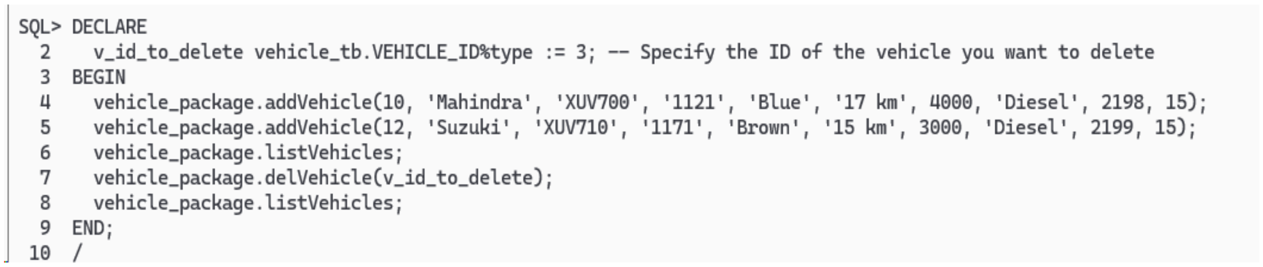
<Package Initialization>
END <package_name>



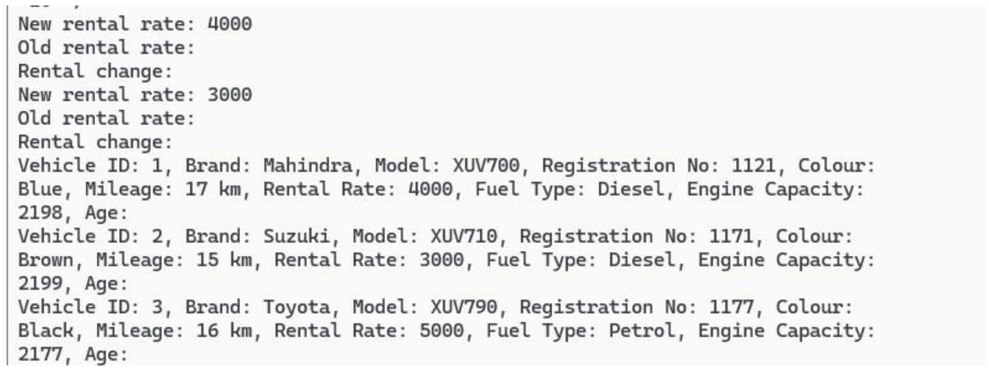
```
SQL> CREATE OR REPLACE PACKAGE vehicle_package AS
    -- Adds a vehicle
    PROCEDURE addVehicle(v_id IN vehicle_tb.VEHICLE_ID%TYPE,
                         v_brand IN vehicle_tb.BRANDNAME%TYPE,
                         v_model IN vehicle_tb.MODEL%TYPE,
                         v_reg_no IN vehicle_tb.REG_NO%TYPE,
                         v_colour IN vehicle_tb.COLOUR%TYPE,
                         v_mileage IN vehicle_tb.MILEAGE%TYPE,
                         v_rental_rate IN vehicle_tb.RENTAL_RATE%TYPE,
                         v_fuel_type IN vehicle_tb.FUEL_TYPE%TYPE,
                         v_engine_capacity IN vehicle_tb.ENGINE_CAPACITY%TYPE,
                         v_age IN vehicle_tb.AGE%TYPE);

    -- Removes a vehicle
    PROCEDURE delVehicle(v_id IN vehicle_tb.VEHICLE_ID%TYPE);

    -- Lists all vehicles
    PROCEDURE listVehicles;
  END vehicle_package;
  /

Package created.
```



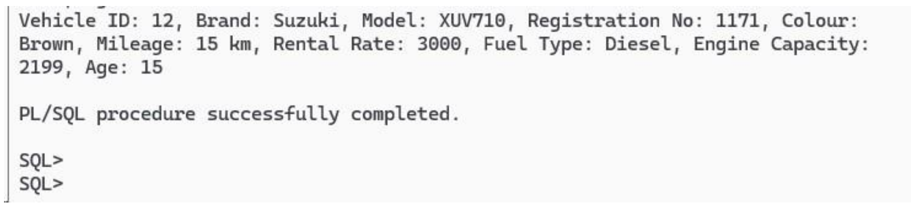
```
SQL> CREATE OR REPLACE PACKAGE BODY vehicle_package AS
    PROCEDURE addVehicle(v_id IN vehicle_tb.VEHICLE_ID%TYPE,
                         v_brand IN vehicle_tb.BRANDNAME%TYPE,
                         v_model IN vehicle_tb.MODEL%TYPE,
                         v_reg_no IN vehicle_tb.REG_NO%TYPE,
                         v_colour IN vehicle_tb.COLOUR%TYPE,
                         v_mileage IN vehicle_tb.MILEAGE%TYPE,
                         v_rental_rate IN vehicle_tb.RENTAL_RATE%TYPE,
                         v_fuel_type IN vehicle_tb.FUEL_TYPE%TYPE,
                         v_engine_capacity IN vehicle_tb.ENGINE_CAPACITY%TYPE,
                         v_age IN vehicle_tb.AGE%TYPE) IS
    BEGIN
      INSERT INTO vehicle_tb (VEHICLE_ID, BRANDNAME, MODEL, REG_NO, COLOUR, MILEAGE, RENTAL_RATE, FUEL_TYPE, ENGINE_CAPACITY, AGE)
      VALUES (v_id, v_brand, v_model, v_reg_no, v_colour, v_mileage, v_rental_rate, v_fuel_type, v_engine_capacity, v_age);
    END addVehicle;

    PROCEDURE delVehicle(v_id IN vehicle_tb.VEHICLE_ID%TYPE) IS
    BEGIN
      DELETE FROM vehicle_tb WHERE VEHICLE_ID = v_id;
    END delVehicle;

    PROCEDURE listVehicles IS
    BEGIN
      FOR vehicle_rec IN (SELECT * FROM vehicle_tb) LOOP
        DBMS_OUTPUT.PUT_LINE('Vehicle ID: ' || vehicle_rec.VEHICLE_ID || ', Brand: ' || vehicle_rec.BRANDNAME ||
                             ', Model: ' || vehicle_rec.MODEL || ', Registration No: ' || vehicle_rec.REG_NO ||
                             ', Colour: ' || vehicle_rec.COLOUR || ', Mileage: ' || vehicle_rec.MILEAGE ||
                             ', Rental Rate: ' || vehicle_rec.RENTAL_RATE || ', Fuel Type: ' || vehicle_rec.FUEL_TYPE ||
                             ', Engine Capacity: ' || vehicle_rec.ENGINE_CAPACITY || ', Age: ' || vehicle_rec.AGE);
      END LOOP;
    END listVehicles;
  END vehicle_package;
  /

Package body created.
```



```
SQL> DECLARE
    v_id_to_delete vehicle_tb.VEHICLE_ID%type := 3; -- Specify the ID of the vehicle you want to delete
  BEGIN
    vehicle_package.addVehicle(10, 'Mahindra', 'XUV700', '1121', 'Blue', '17 km', 4000, 'Diesel', 2198, 15);
    vehicle_package.addVehicle(12, 'Suzuki', 'XUV710', '1171', 'Brown', '15 km', 3000, 'Diesel', 2199, 15);
    vehicle_package.listVehicles;
    vehicle_package.delVehicle(v_id_to_delete);
    vehicle_package.listVehicles;
  END;
  /
```



```
New rental rate: 4000
Old rental rate:
Rental change:
New rental rate: 3000
Old rental rate:
Rental change:
Vehicle ID: 1, Brand: Mahindra, Model: XUV700, Registration No: 1121, Colour:
Blue, Mileage: 17 km, Rental Rate: 4000, Fuel Type: Diesel, Engine Capacity:
2198, Age:
Vehicle ID: 2, Brand: Suzuki, Model: XUV710, Registration No: 1171, Colour:
Brown, Mileage: 15 km, Rental Rate: 3000, Fuel Type: Diesel, Engine Capacity:
2199, Age:
Vehicle ID: 3, Brand: Toyota, Model: XUV790, Registration No: 1177, Colour:
Black, Mileage: 16 km, Rental Rate: 5000, Fuel Type: Petrol, Engine Capacity:
2177, Age:
```



```
Vehicle ID: 12, Brand: Suzuki, Model: XUV710, Registration No: 1171, Colour:
Brown, Mileage: 15 km, Rental Rate: 3000, Fuel Type: Diesel, Engine Capacity:
2199, Age: 15

PL/SQL procedure successfully completed.

SQL>
SQL>
```

<u>Result:</u> Thus the functions , procedures and packages are successfully executed in SQL