Database Systems Lab - 14CS2012

REGISTER NO: UR14CS228

DATE: 31-10-16

EXPERIMENT-NO 8

Video Link : https://youtu.be/Szpl-eIcNG4

AIM:
To Create Functions and Procedures for the following requirements.

DESCRIPTION:

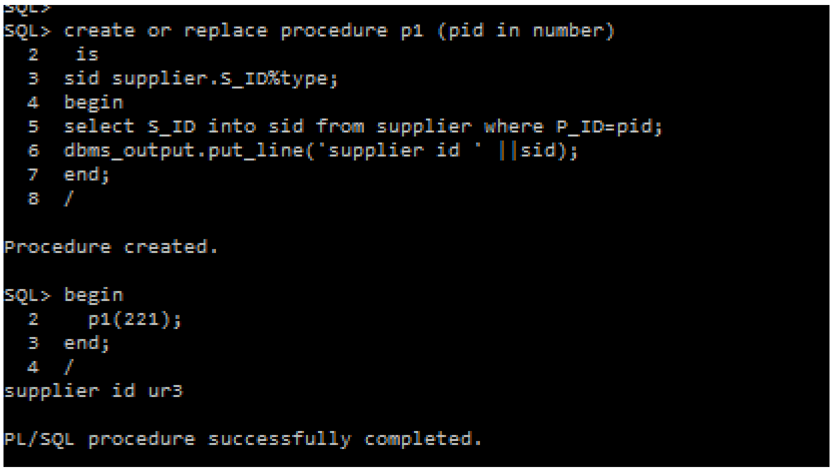
PL/SQL subprograms are named PL/SQL blocks that can be invoked with a set of parameters. PL/SQL provides two kinds of subprograms:

- **Functions**: these subprograms return a single value, mainly used to compute and return a value.
- **Procedures**: these subprograms do not return a value directly, mainly used to perform an action.

Program:

1. Write a PL/SQL procedure that will accept the product id from the user, check if the product is supplied by the supplier and display the status.

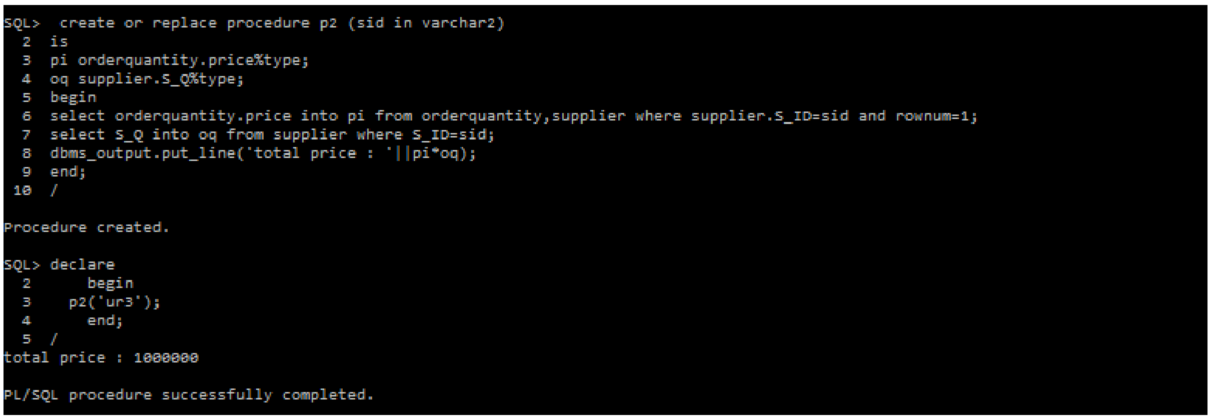
```
create or replace procedure p1 (pid in number) is
sid supplier.S_ID%type;
begin
select S_ID into sid from supplier where P_ID=pid;
dbms_output.put_line('supplier id ' ||sid);
end;


begin
  p1(221);
end;
```



```
SQL> create or replace procedure p1 (pid in number)
      is
     sid supplier.S_ID%type;
     begin
     select S_ID into sid from supplier where P_ID=pid;
     dbms_output.put_line('supplier id ' ||sid);
     end;
     /

Procedure created.

SQL> begin
       p1(221);
     end;
     /
supplier id ur3

PL/SQL procedure successfully completed.
```

2. Write a procedure to calculate total price for the product that has been supplied and Pass the supplier id as the argument.

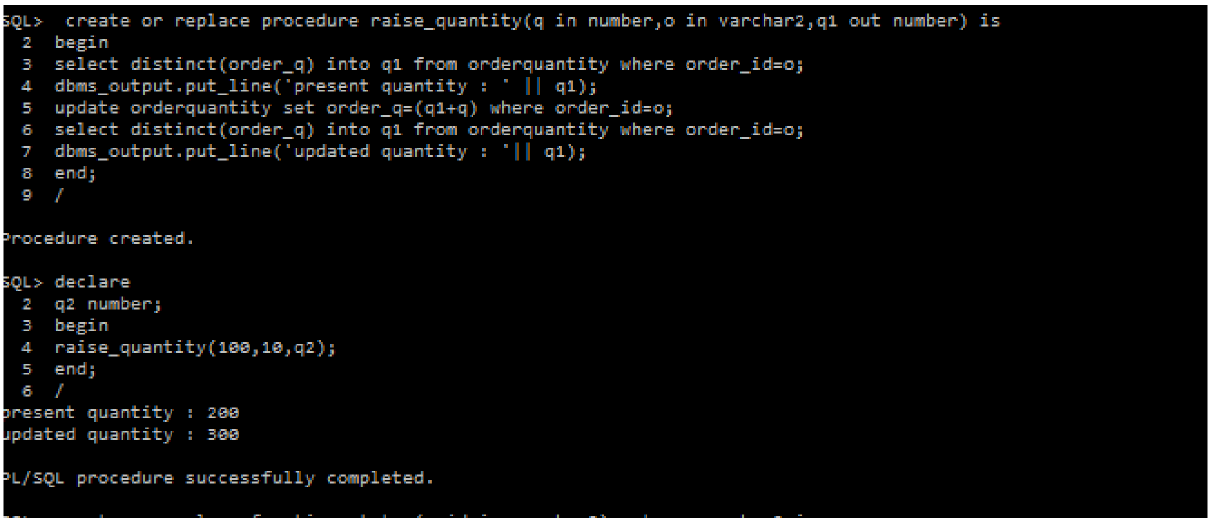
```
create or replace procedure p2 (sid in varchar2)

is

pi orderquantity.price%type;

oq supplier.S_Q%type;

begin

select orderquantity.price into pi from orderquantity,supplier
where supplier.S_ID=sid and rownum=1;

select S_Q into oq from supplier where S_ID=sid;

dbms_output.put_line('total price : '||pi*oq);

end;


declare

    begin

  p2('ur3');

    end;
```



```
SQL>  create or replace procedure p2 (sid in varchar2)
     is
     pi orderquantity.price%type;
     oq supplier.S_Q%type;
     begin
     select orderquantity.price into pi from orderquantity,supplier where supplier.S_ID=sid and rownum=1;
     select S_Q into oq from supplier where S_ID=sid;
     dbms_output.put_line('total price : '||pi*oq);
     end;
     /

Procedure created.

SQL> declare
         begin
       p2('ur3');
         end;
     /
total price : 1000000

PL/SQL procedure successfully completed.
```

3. Write a procedure raise_quantity which increases the quantity of the ordered product. It accepts an order id and quantity to be increased. It uses the order id to find the current quantity from the ORDER table and update the quantity.

```
create or replace procedure raise_quantity(q in number,o in varchar2,q1 out number) is

begin

select distinct(order_q) into q1 from orderquantity where order_id=o;

dbms_output.put_line('present quantity : ' || q1);

update orderquantity set order_q=(q1+q) where order_id=o;

select distinct(order_q) into q1 from orderquantity where order_id=o;

dbms_output.put_line('updated quantity : '|| q1);

end;


declare

q2 number;

begin

raise_quantity(100,10,q2);

end;
```

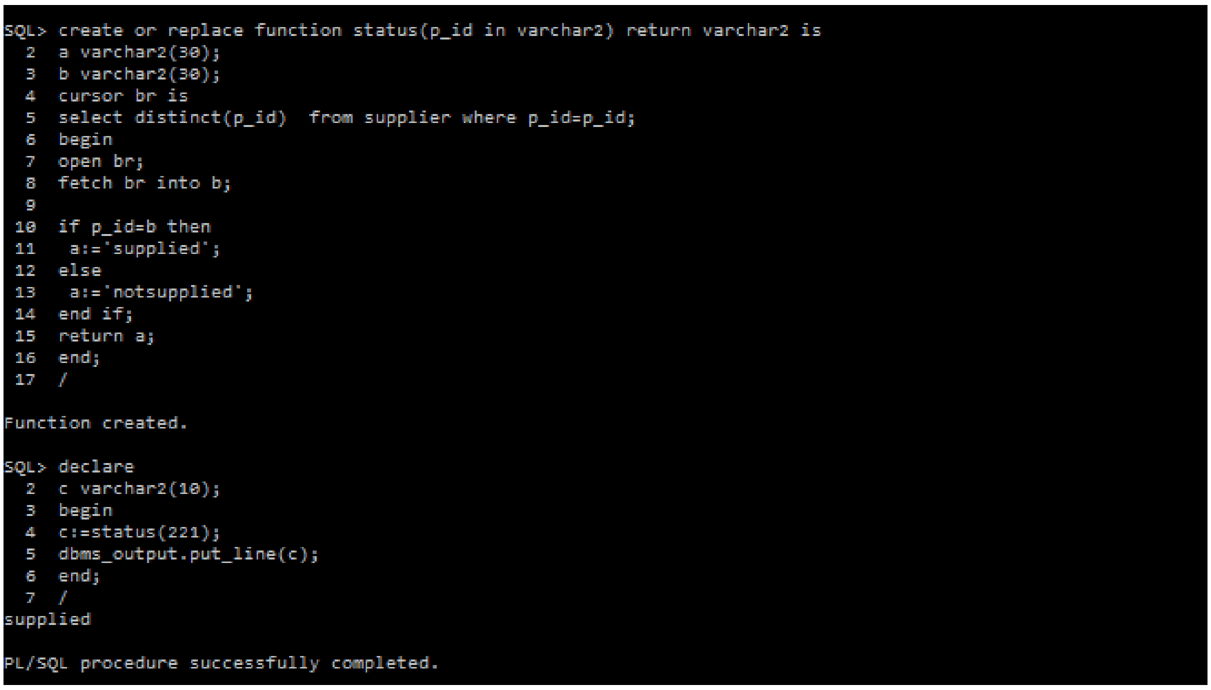


```
SQL>  create or replace procedure raise_quantity(q in number,o in varchar2,q1 out number) is
 begin
 select distinct(order_q) into q1 from orderquantity where order_id=o;
 dbms_output.put_line('present quantity : ' || q1);
 update orderquantity set order_q=(q1+q) where order_id=o;
 select distinct(order_q) into q1 from orderquantity where order_id=o;
 dbms_output.put_line('updated quantity : '|| q1);
 end;
 /

Procedure created.

SQL> declare
 q2 number;
 begin
 raise_quantity(100,10,q2);
 end;
 /
present quantity : 200
updated quantity : 300

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL function STATUS to return value SUPPLIED if the product number passed to it is available in the Supplier table else will return NOT SUPPLIED.

```
create or replace function status(p_id in varchar2) return varchar2 is

a varchar2(30);

b varchar2(30);
```

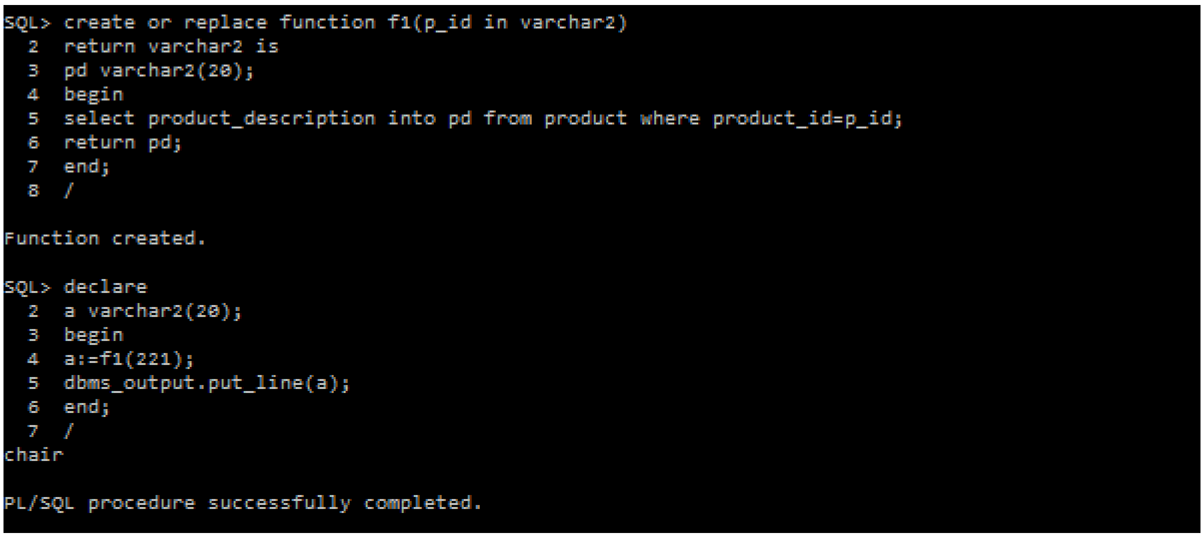
```
cursor br is
select distinct(p_id)  from supplier where p_id=p_id;
begin
open br;
fetch br into b;

if p_id=b then
 a:='supplied';
else
 a:='notsupplied';
end if;
return a;
end;

declare
c varchar2(100);
begin
c:=status(21);
dbms_output.put_line(c);
end;
```



```
SQL> create or replace function status(p_id in varchar2) return varchar2 is
a varchar2(30);
b varchar2(30);
cursor br is
select distinct(p_id)  from supplier where p_id=p_id;
begin
open br;
fetch br into b;

if p_id=b then
 a:='supplied';
else
 a:='notsupplied';
end if;
return a;
end;
/

Function created.

SQL> declare
c varchar2(10);
begin
c:=status(221);
dbms_output.put_line(c);
end;
/
supplied

PL/SQL procedure successfully completed.
```

5. Write a PL/SQL function to return the product name when the product id is passed as an argument.

```
create or replace function f1(p_id in varchar2)

return varchar2 is

pd varchar2(20);

begin

select product_description into pd from product where
product_id=p_id;

return pd;

end;


declare

a varchar2(20);

begin

a:=f1(221);

dbms_output.put_line(a);

end;
```

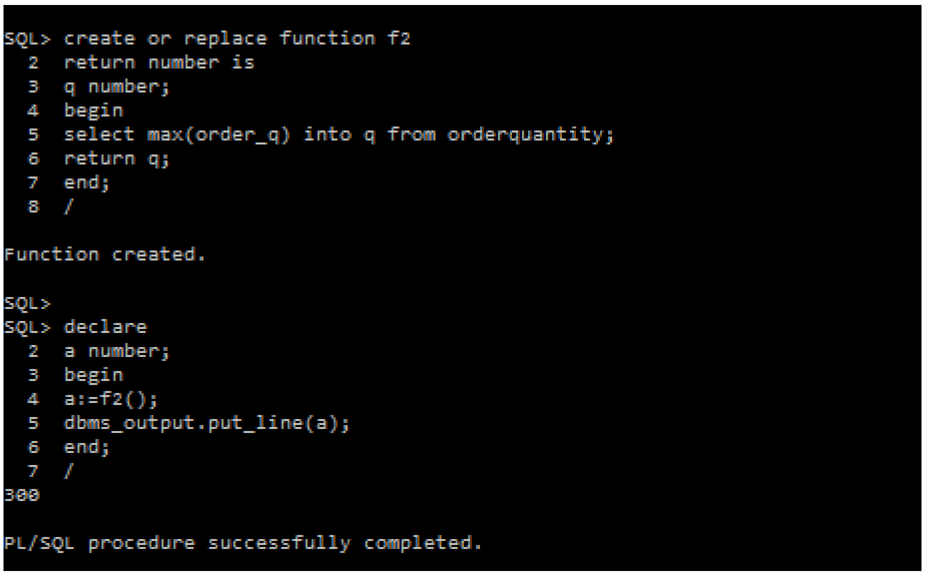


```
SQL> create or replace function f1(p_id in varchar2)
  2  return varchar2 is
  3  pd varchar2(20);
  4  begin
  5  select product_description into pd from product where product_id=p_id;
  6  return pd;
  7  end;
  8  /

Function created.

SQL> declare
  2  a varchar2(20);
  3  begin
  4  a:=f1(221);
  5  dbms_output.put_line(a);
  6  end;
  7  /
chair

PL/SQL procedure successfully completed.
```

6. Write a PL/SQL function to return the maximum quantity ordered by the customers.

```
create or replace function f2

return number is

q number;

begin

select max(order_q) into q from orderquantity;

return q;
```

```
end;


declare
a number;
begin
a:=f2();
dbms_output.put_line(a);
end;
```



```
SQL> create or replace function f2
  2  return number is
  3  q number;
  4  begin
  5  select max(order_q) into q from orderquantity;
  6  return q;
  7  end;
  8  /

Function created.

SQL>
SQL> declare
  2  a number;
  3  begin
  4  a:=f2();
  5  dbms_output.put_line(a);
  6  end;
  7  /
300

PL/SQL procedure successfully completed.
```

Result:

Functions and procedures were successfully created and tested for all the different situations