Procedural language/Structured query language(PL/SQL)

Aim: To study PL/SQL

Theory:

Although SQL is a very powerful tool, it does not have any procedural capabilities like conditional checking, looping, branching etc. For a fully structured programming language Oracle provides PL/SQL. PL/SQL is a superset of SQL.

Syntax:

Declare (declarations of memory variables, constants etc)
Begin(SQL and PL/SQL executable statements)
Exception(built-in and user defined exceptions)
End(end of PL/SQL block)

Displaying user message on screen:

SQL>set serveroutput on

DBMS_OUTPUT.PUT_LINE('message');

DBMS_OUTPUT is a package that includes a number of procedures and functions that accumulate information in a buffer so that it can be retrieved later.

PUT_LINE is a function to display message that excepts single parameter of character data type.

Exercise

SQL>set serveroutput on

Q1. Display the numbers from 1 to 10 in ascending order.

```
begin
declare
a number;

for a in 1..10 loop
dbms_output.put_line(to_char(a));
end loop;
end;
```

Q2. Display the numbers from 1 to 10 in descending order.

```
begin
declare
a number;

for a in reverse 1..10 loop
dbms_output.put_line(to_char(a));
end loop;
end;
```

Q3. Print the given number in reverse order.

```
begin
declare
no varchar(4):='1234';
len number(2);
rev varchar(4);
cnt number(2);

len:=length(no);
for cnt in reverse 1.. len loop
rev:=rev||substr(no,cnt,1);
end loop;
dbms_output.put_line('The given number is'||no);
dbms_output.put_line('The reverse number is'||rev);
end;
```

Q4. Read the title of the book from the user and update the number of copies in Qty_in_stock.

```
begin
declare
 a number;
 b varchar(10);
dbms_output.put_line('Enter The Title:');
 b:='&TITLE';
 dbms_output.put_line(to_char(b));
 dbms_output.put_line('Enter the number of copies');
 a:='&QTY';
 dbms_output.put_line(to_char(a));
 update book set QTY=a where TITLE=b;
 commit;
 end;
```

Q5. Take as input the title of book and quantity of the same bought. Look up the 'book' table to get price and if copies are less than 10 give 20% discount, if between 10 and 20 give 40% discount else 50% discount. Display total price after discount.

```
declare
a number;
b varchar(10);
c number;
d number;
e number(10,2);
begin
dbms_output.put_line('Enter The Title:');
b:='&TITLE';
dbms_output.put_line('Enter the number of copies');
a:='&QTY';
select price into e from book where TITLE=b;
if a < 10 then
c:=0.2*e;
elsif a between 10 and 20 then
c:=0.4*e;
else
c:=0.5*e;
end if;
d := e–c;
dbms_output.put_line('The total price = '||d);
end;
```

Q6. Write a PL/SQL block that takes book isbn and customer ID as input and displays fine to be issued for the corresponding order (in OrderBook table). Fine is 25% of the book price if the book was returned more than a week after the book was ordered. Consider system date as date of return.

```
declare
pr number;
fine number; b
varchar(4);
doi date;
id varchar(4);
begin fine:=0;
b:='&isbn';
dbms_output.put_line(to_char(b));
id:='&ocid';
dbms_output.put_line(to_char(id));
select price into pr from book where isbn=b;
select order_date into doi from order1 where oisbn=b and ocid=id;
if((to_date(sysdate,'dd-mm-yy'))-(to_date(doi,'dd-mm-yy')))>7 then
fine:=pr*25/100;
end if;
dbms_output.put_line('Fine is'||fine);
end;
```

Conclusion:

Studied various control statements like conditional, iterative and decision control in PL/SQL programming.