PL/SQL - Notes Part 2 for students References

Cursors
****
Oracle creates a memory area, known as context area, for processing an SQL statement,
which contains all information needed for processing the statement,
for example, number of rows processed etc.

Implicit Cursors
****
Implicit cursors are automatically created by Oracle whenever an SQL statement is executed, when there is no explicit cursor for the statement.

Programmers cannot control the implicit cursors and the information in it.

Implicit cursors will process 1 records in a table.

Implicit Cursor attributes.
****
%FOUND will aways return true if insert, update or delete is sucusseful.
    else it returns false.

%NOTFOUND it will true statement if insert , update or delete is not
    sucessful else it return false.

%ISOPEN
%ROWCOUNT

access the attributes with the following syntax
sql%attribute_name
for example sql%rowcount

==
==

Implicit cursor programs.
****

/*
a pl/sql block using implicit cursor where commission is decreased by Rs. 200 for
all SALESPEOPLE AND BLOCK WILL display how many person insentive decreased
if the query is sucessfull.
*/
Set serveroutput on;

```
DECLARE
 total_rows number(4);

BEGIN

   UPDATE salespeople SET comm = comm - 
200 ;

   IF  sql%found
      THEN
          total_rows := sql%rowcount;
          dbms_output.put_line( total_rows || '
Salespeople Insentive  Decreased');

          commit;

   END IF;
END;
/
```

=======================================
=====================
=======================================
=====================

Explicit cursor
****

****
Explicit cursors are programmer defined cursors for gaining more control over the context area.
An explicit cursor should be defined in the declaration section of the PL/SQL Block.
It is created on a SELECT Statement which returns more than one row.

Explicit cursor has to be declared in the declare section, cursor has to be opend and records fetched from cursor and then cursor has to be closed in the end.

attributes.
****
%found
%notfound
%isopen
%rowcount

=====================================
=============

/*
write a pl/sql block using explicit cursors to declare a explict cursor fetch all the
customer table tupples and print the report using selected attrributes.

```
*/

set serveroutput on;
DECLARE
   c_id customers.cnum%type;
   c_name customers.cname%type;
   c_addr customers.city%type;

   CURSOR c_customers is
      SELECT cnum, cname, city FROM
customers;

BEGIN

   OPEN c_customers;

  LOOP
      FETCH c_customers into c_id, c_name,
c_addr;
          EXIT WHEN c_customers%notfound;

      dbms_output.put_line(c_id || ' ' || c_name || ' '
|| c_addr);

 END LOOP;

  CLOSE c_customers;
```

```
END;
/
```

====================================
====================
====================================
====================

explicit cursor using for loop
****

write a pl/sql block to print all customers staying in Bengaluru
using for loop only.

```
set serveroutput on;

DECLARE
  CURSOR cus IS SELECT  cnum, cname, city
FROM customers
                              where city = 'London' or city
= 'Bengaluru'
                              order by cname desc;

BEGIN
        FOR r in cus
               LOOP
                  DBMS_OUTPUT.PUT_LINE('cnum
is ' || r.cnum);

DBMS_OUTPUT.PUT_LINE('Customer name is '
|| r.cname);
                DBMS_OUTPUT.PUT_LINE(' City is
' || r.city);
          END LOOP;
END;
/
```

=====================================
====================
=====================================
====================

Exceptions
****
An error condition during a program execution is called an exception in PL/SQL.

PL/SQL supports programmers to catch such conditions using EXCEPTION block in
the program and an appropriate action is taken against the error condition.

/*
Exception examples.
****

```
write a pl/sql block to fetch the details of
customer number 88
use the in built exceptions to print "CUsotmer not
found if the cutomer no does not exist."
*/

set serveroutput on;

DECLARE
   c_id customers.cnum%type := 88;
   c_name  customers.cname%type;
   c_addr customers.city%type;
BEGIN
   SELECT  cname, city INTO  c_name, c_addr
   FROM customers
   WHERE cnum = c_id;

   DBMS_OUTPUT.PUT_LINE ('Name: '||
c_name);
   DBMS_OUTPUT.PUT_LINE ('Address: ' ||
c_addr);

EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('This customers no
does not exist in  table!');

   WHEN others THEN
```

```
    dbms_output.put_line('some other Error!');
END;
/
```

==
=====================
==
=====================

PL/SQL – Procedures
****

A subprogram is a program unit/module that performs a particular task.

These subprograms are combined to form larger programs. This is basically called the 'Modular design'. A subprogram can be invoked by another subprogram or program, which is called the calling program.

examples of creating a stand alone Procedure
****
*

```
sql>
CREATE OR REPLACE PROCEDURE greetings
AS
```

```
BEGIN
   dbms_output.put_line('Welcome to the world of
PL/SQL Programming');
END;
/
```

to execute the above procedure

```
sql>execute greetings;
```

==
====================
==
====================

write a pl/sql block where you will declare a procedure within a block called findMin which will

receive 2 variables values and return the lowest of 2 numbers.
****
****

```
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:= &a;
   b:= &b;

   findMin(a, b, c);
   dbms_output.put_line(' Minimum of  both
numbers is ' || c);
END;
/
```

==
====================
==
====================

to see the user defined procedures
****

To list all stored procedures in the database you're connected to

```
sql> select object_name from user_procedures;
```

==

To list stand alone procedures in the database you're connected to

```
SQL> select object_name from user_procedures
          where object_name = 'GREETINGS';
```

==

Functions
****

Creating a Function
A standalone function is created using the CREATE FUNCTION statement.

Following stand alone function will print the number of records from a customer table.
****

****

```
CREATE OR REPLACE FUNCTION
totalCustomers
RETURN number IS
   total number(4) := 0;
BEGIN
   SELECT count(*) into total
   FROM customers;

   RETURN total;
END;
/
```

===
===================

The above function can be called from the following pl/sql block.
****
****

```
set serveroutput on;

DECLARE
   c number(4);

BEGIN
```

```
   c := totalCustomers();

   dbms_output.put_line('Total no  of Customers
is : ' || c);
END;
/
==========================================
===================
==========================================
===================
```

to list all the functions in PL/SQL
****

```
sql> select object_name  from user_objects
where object_type = 'FUNCTION';
```

==
==================